Least Mean Squares Regression

Machine Learning

THE
UNIVERSITY
OF UTAH

Least Squares Method for regression

- Examples
- The LMS objective
- Gradient descent
- Incremental/stochastic gradient descent

Least Squares Method for regression

- Examples
- The LMS objective
- Gradient descent
- Incremental/stochastic gradient descent

What's the mileage?

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

`Weight` (x 100 lb) x_1	`Age` (years) x_2	`Mileage`
31.5	6	21
36.2	2	25
43.1	0	18
27.6	2	30

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

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

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

Predicting continuous values using a linear model

Assumption: The output is a linear function of the inputs

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

Parameters of the model
Also called weights
Collectively, a vector

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: $\mathbf{x} \in \Re^d$
- Outputs are real numbers: $y \in \Re$

- We have a training set

$$\mathrm{D} = \{\, (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots \}$$

- We want to approximate y as

$$y = w_1 + w_2 x_2 + \cdots + w_d x_d$$

$$y = \mathbf{w}^T \mathbf{x}$$

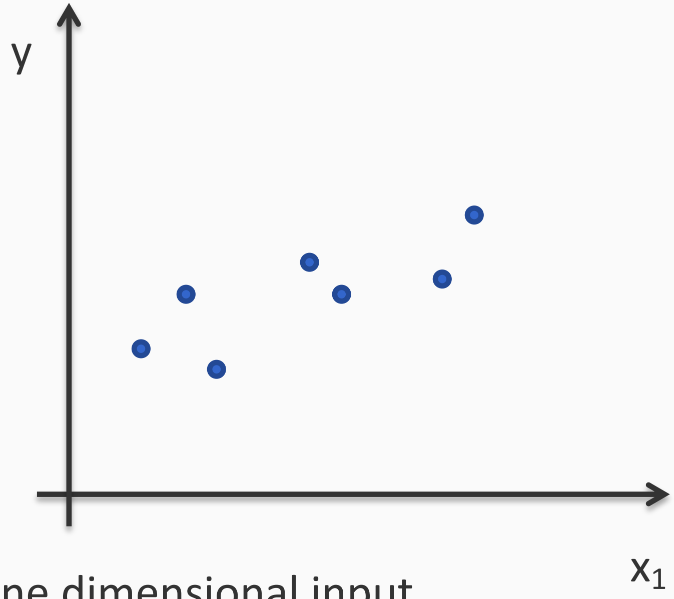
$\mathbf{w}$ is the learned weight vector in $\Re^d$

For simplicity, we will assume that the first feature is always 1.

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

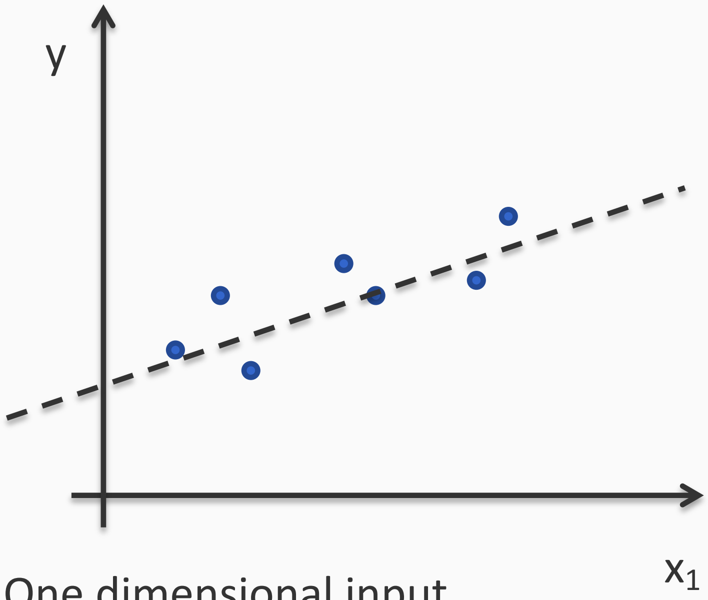
This lets makes notation easier

Examples



One dimensional input

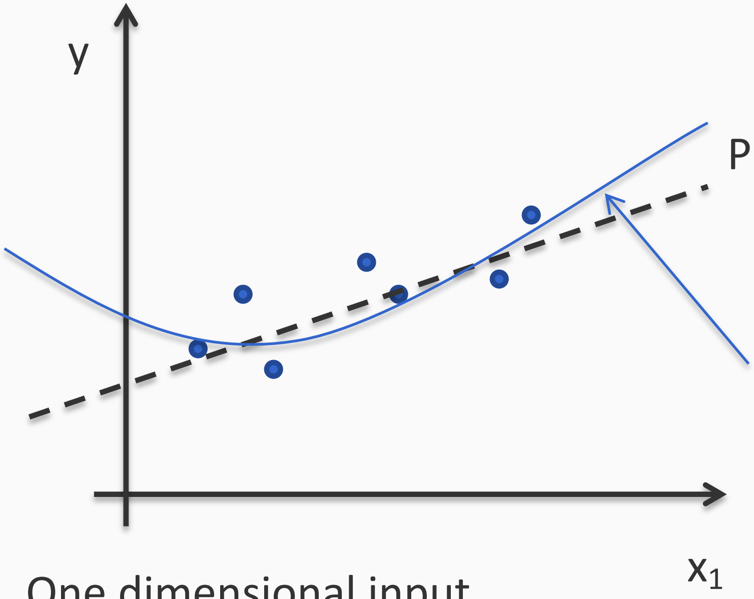
Examples



Predict using $y = w_1 + w_2 x_2$

One dimensional input

Examples

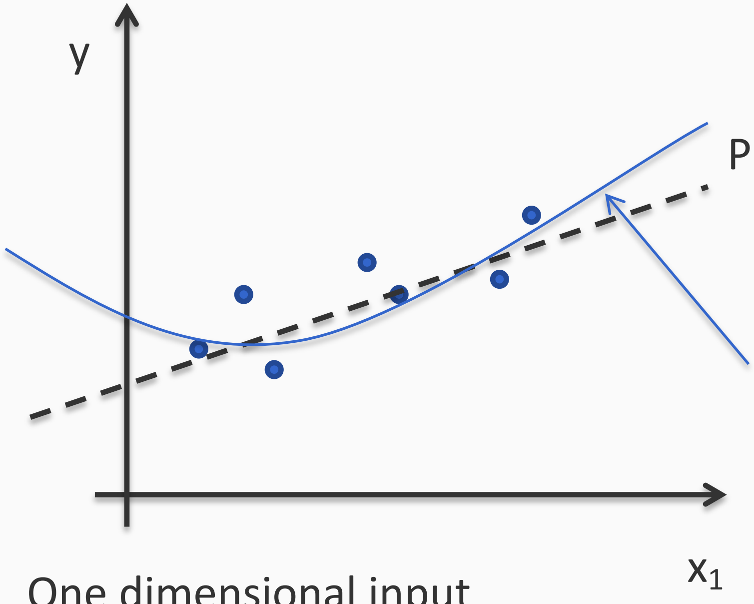


Predict using $y = w_1 + w_2\, x_2$

The linear function is not our only choice. We could have tried to fit the data as another polynomial

One dimensional input

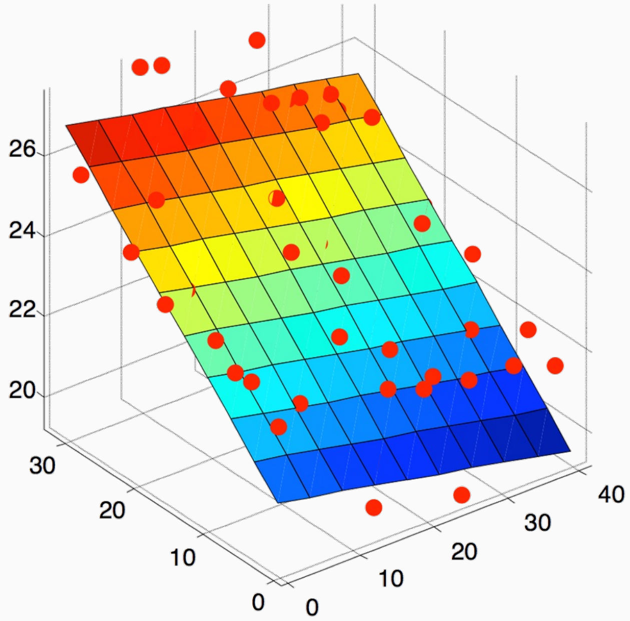
Examples



Predict using $y = w_1 + w_2 x_2$

The linear function is not our only choice. We could have tried to fit the data as another polynomial

One dimensional input



Two dimensional input

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

Least Squares Method for regression

- Examples
- The LMS objective
- Gradient descent
- Incremental/stochastic gradient descent

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 input ($\mathbf{x}_i$, y_i) in the training set, the ***cost*** of a mistake is

$$\left|y_i - \mathbf{w}^T\mathbf{x}_i\right|$$

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 ($\mathbf{x}_i$, y_i) in the training set, the ***cost*** of a mistake is

$$\left|y_i - \mathbf{w}^T\mathbf{x}_i\right|$$

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

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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 ($\mathbf{x}_i$, y_i) in the training set, the ***cost*** of a mistake is

$$\left| y_i - \mathbf{w}^T \mathbf{x}_i \right|$$

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

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{m} \left(y_i - \mathbf{w}^T \mathbf{x}_i \right)^2$$

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 ($\mathbf{x}_i$, y_i) in the training set, the ***cost*** of a mistake is

$$\left|y_i - \mathbf{w}^T\mathbf{x}_i\right|$$

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

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Sum of squared costs over the training set

One strategy for learning: *Find the **w** with least cost on this data*

Least Mean Squares (LMS) Regression

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

Learning: minimizing mean squared error

Least Mean Squares (LMS) Regression

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

Learning: minimizing mean squared error

Different strategies exist for *learning by optimization*

- Gradient descent is a popular algorithm

  (For this particular minimization objective, there is also an analytical solution. No need for gradient descent)

Least Squares Method for regression

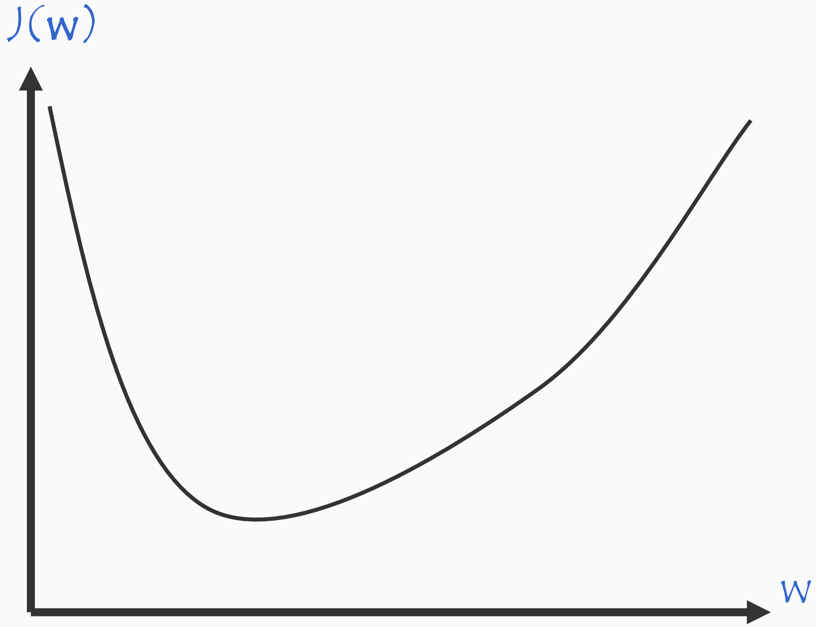
- Examples
- The LMS objective
- Gradient descent
- Incremental/stochastic gradient descent

Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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



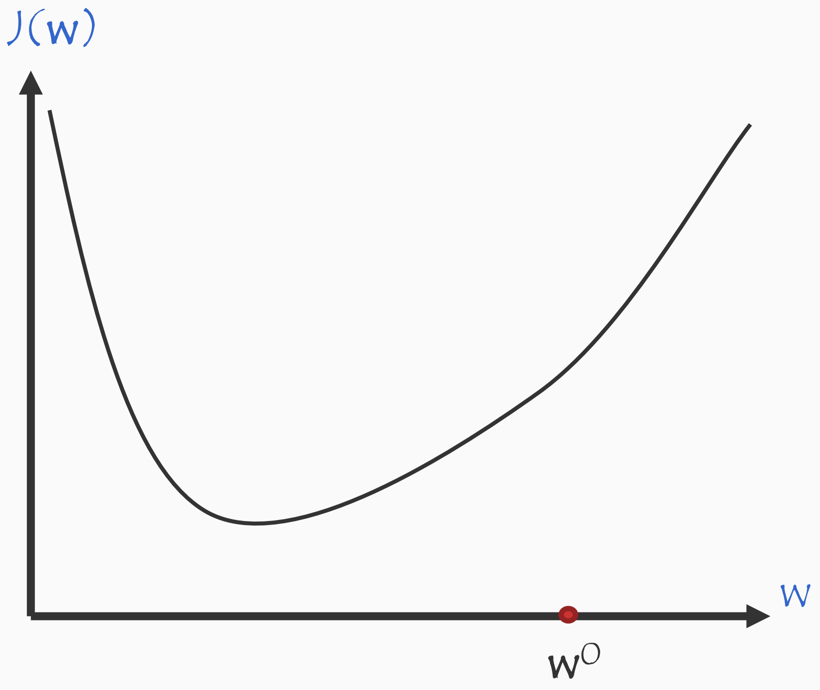
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

- Start with an initial guess for **w**, say $\mathbf{w}^0$



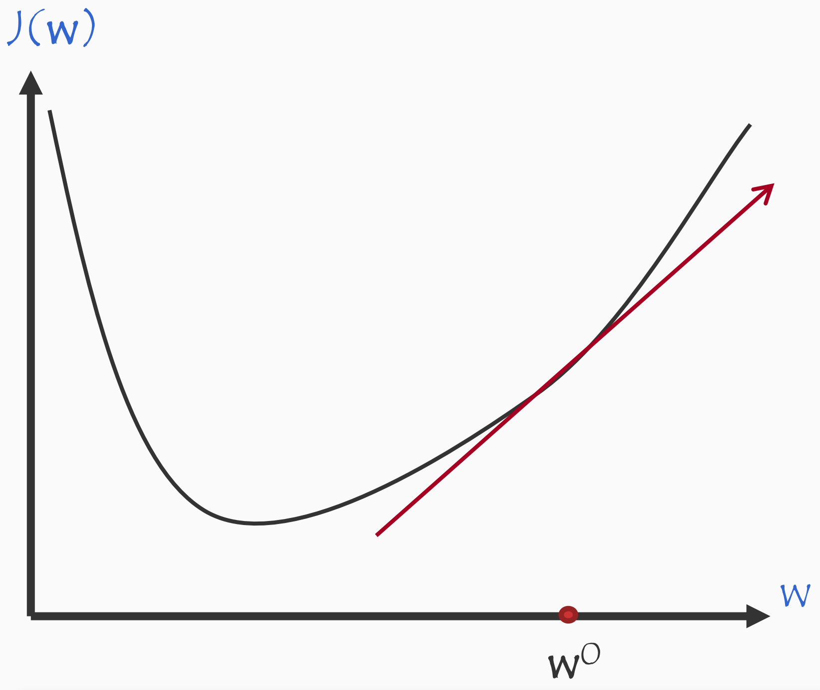
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

- Start with an initial guess for **w**, say $\mathbf{w}^0$



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

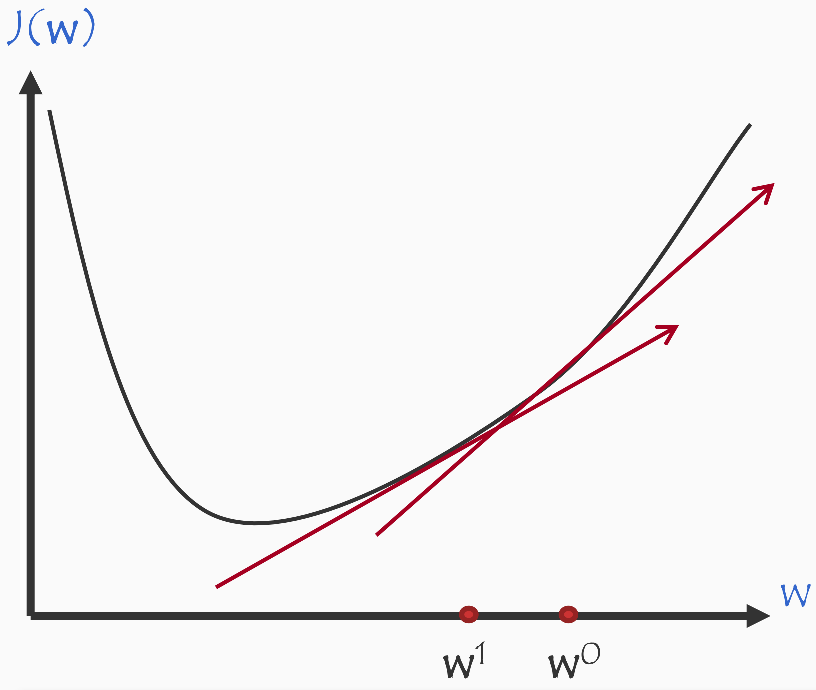
Gradient descent

We are trying to minimize

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

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

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



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

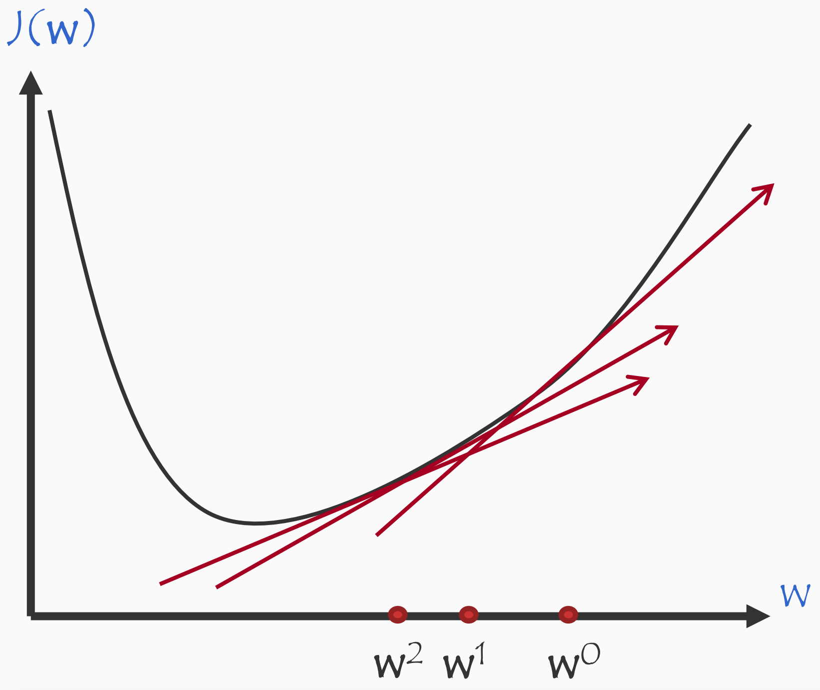
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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



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

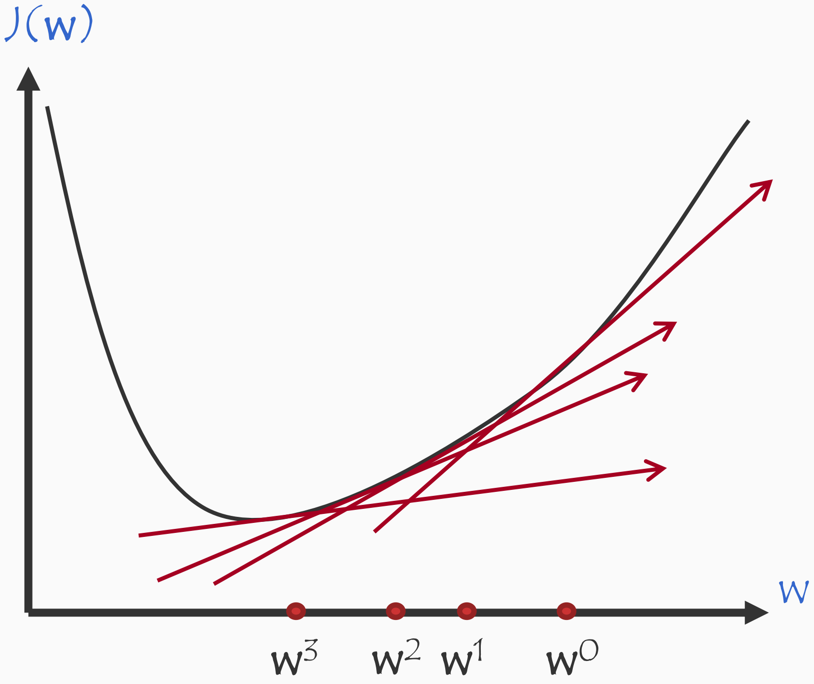
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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



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

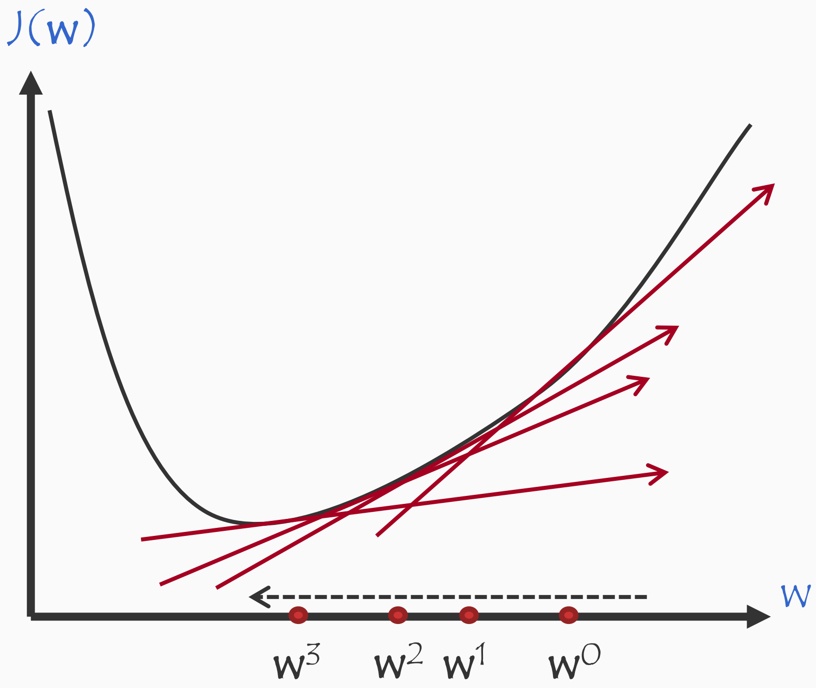
Gradient descent

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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



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

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of $J(\mathbf{w}^t)$ at $\mathbf{w}^t$. Call it $\nabla J(\mathbf{w}^t)$
   2. Update w as follows:

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

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

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….

   What is the gradient of J?

   1. Compute gradient of $J(\mathbf{w}^t)$ at $\mathbf{w}^t$. Call it $\nabla J(\mathbf{w}^t)$
   2. Update w as follows:

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

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

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Gradient of the cost

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

- Remember that **w** is a vector with d elements
  - **w** = $[w_1, w_2, w_3, \cdots w_j, \cdots, w_d]$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Gradient of the cost

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

$$\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Gradient of the cost

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

$$\begin{aligned}\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2\end{aligned}$$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Gradient of the cost

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

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}\left(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots\right)
\end{aligned}$$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

Gradient of the cost

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

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}\left(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots\right) \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)(-x_{ij})
\end{aligned}$$

Gradient of the cost

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}\left(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots\right) \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)(-x_{ij}) \\
&= -\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}
\end{aligned}$$

Gradient of the cost

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots) \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)(-x_{ij}) \\
&= \boxed{-\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}}
\end{aligned}$$

One element of the gradient vector

Gradient of the cost

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

$$
\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)\frac{\partial}{\partial w_j}(y_i - w_1x_{i1} - \cdots w_jx_{ij} - \cdots) \\
&= \frac{1}{2}\sum_{i=1}^{m}2(y_i - \mathbf{w}^T\mathbf{x}_i)(-x_{ij}) \\
&= \boxed{-\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}}
\end{aligned}
$$

Sum of   Error × Input

One element of the gradient vector

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
   1. Compute gradient of J(**w**) at $\mathbf{w}^t$. Call it $\nabla J(w^t)$

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

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

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
    1. Compute gradient of J(**w**) at $\mathbf{w}^t$. Call it $\nabla J(w^t)$

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

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

One element of $\nabla J(w^t)$

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, ….
    1. Compute gradient of J(**w**) at $\mathbf{w}^t$. Call it $\nabla J(w^t)$

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

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

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

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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

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

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

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

1. Initialize $\mathbf{w}^0$
2. For t = 0, 1, 2, …. *(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

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

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

       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

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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

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

       One element of $\nabla J(w^t)$

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

       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

Least Squares Method for regression

- Examples
- The LMS objective
- Gradient descent
- Incremental/stochastic gradient descent

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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

       $$\frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}$$
    2. Update **w** as follows: $\mathbf{w}^{t+1} = \mathbf{w}^t - r\nabla J(\mathbf{w}^t)$

Gradient descent for LMS

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$$

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

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

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

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

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?

Incremental/Stochastic gradient descent

- Repeat for each example ($\mathbf{x}_i$, 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

Incremental/Stochastic gradient descent

1. Initialize **w**
2. For t = 0, 1, 2, …. (until error below some threshold)
   - For each training example ($\mathbf{x}_i$, y_i):
     - Update **w**. For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i)x_{ij}$$

Incremental/Stochastic gradient descent

1. Initialize **w**
2. For t = 0, 1, 2, …. (until error below some threshold)
   - For each training example ($\mathbf{x}_i$, y_i):
     - Update **w**. For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T\mathbf{x}_i)x_{ij}$$

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

Incremental/Stochastic gradient descent

1. Initialize **w**
2. For t = 0, 1, 2, …. (until error below some threshold)
   - For each training example ($\mathbf{x}_i$, y_i):
     - Update **w**. For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}$$

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

Incremental/Stochastic gradient descent

1. Initialize **w**
2. For t = 0, 1, 2, …. (until error below some threshold)
   - For each training example ($\mathbf{x}_i$, y_i):
     - Update **w**. For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}$$

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

Online/Incremental 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 (non-separable) 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. Use the gradient descent algorithms to solve the mileage problem (on paper, or write a small program)

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

$$X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \mathbf{x}_m \end{bmatrix}_{d \times m} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1}$$

Show that the optimization problem we saw earlier is equivalent to

$$\min_{\mathbf{w}} \left(X^T\mathbf{w} - Y\right)^T \left(X^T\mathbf{w} - Y\right)$$

This can be solved analytically. Show that the solution w* is

$$\mathbf{w}^* = \left(XX^T\right)^{-1} XY$$

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