Introduction to Artificial Intelligence

Optimization

Andres Mendez-Vazquez

January 23, 2020

Outline

Outline

At the beginning

We have have

- Fermat
  - "Methodus ad disquirendam maximam et minimam et de tangentibus linearum curvarum"
- Lagrange
  - He was one of the creators of the calculus of variations, deriving the Euler–Lagrange equations for extrema of functionals.

At the beginning

We have have

- Fermat
  - "Methodus ad disquirendam maximam et minimam et de tangentibus linearum curvarum"
- Lagrange
  - He was one of the creators of the calculus of variations, deriving the Euler–Lagrange equations for extrema of functionals.

Then

- Newton and Gauss
  - They proposed methods for moving towards an optimum.

At the beginning

We have have

- Fermat
  - "Methodus ad disquirendam maximam et minimam et de tangentibus linearum curvarum"
- Lagrange
  - He was one of the creators of the calculus of variations, deriving the Euler–Lagrange equations for extrema of functionals.

Then

- Newton and Gauss
  - They proposed methods for moving towards an optimum.

At the beginning

We have have

- Fermat
  - "Methodus ad disquirendam maximam et minimam et de tangentibus linearum curvarum"
- Lagrange
  - He was one of the creators of the calculus of variations, deriving the Euler–Lagrange equations for extrema of functionals.

Then

- Newton and Gauss
  - They proposed methods for moving towards an optimum.

But it is until the 20^{th} century

We have

- Leonid Kantorovich - Nobel Memorial Prize in Economic Sciences
  - Developed much of what is know as linear programming
- Dantzig
  - He published the Simplex algorithm in 1947
- John von Neumann in 1947
  - Developed the min-max
  - And the Theory of Duality

But it is until the 20^{th} century

We have

- Leonid Kantorovich - Nobel Memorial Prize in Economic Sciences
  - Developed much of what is know as linear programming
- Dantzig
  - He published the Simplex algorithm in 1947
- John von Neumann in 1947
  - Developed the min-max
  - And the Theory of Duality

But it is until the 20^{th} century

We have

- Leonid Kantorovich - Nobel Memorial Prize in Economic Sciences
  - Developed much of what is know as linear programming
- Dantzig
  - He published the Simplex algorithm in 1947
- John von Neumann in 1947
  - Developed the min-max
  - And the Theory of Duality

Outline

Definition of the Problem

A mathematical optimization problem

$$minimize f_0(\boldsymbol{x})$$
$$s.t. f_i(\boldsymbol{x}) \leq b_i \; i = 1, ..., m$$

Here, the vector $x^T = (x_1, ..., x_n)$

It is the optimization variable of the problem problem

The function $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$

It is the objective function.

Definition of the Problem

A mathematical optimization problem

$$minimize f_0(\boldsymbol{x})$$
$$s.t. f_i(\boldsymbol{x}) \leq b_i \; i = 1, ..., m$$

Here, the vector $\boldsymbol{x}^T = (x_1, ..., x_n)$

It is the optimization variable of the problem problem

The function $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$

It is the objective function.

Definition of the Problem

A mathematical optimization problem

$$
\begin{aligned}
minimize f_0(\boldsymbol{x}) \\
s.t. f_i(\boldsymbol{x}) \leq b_i \ i = 1, ..., m
\end{aligned}
$$

Here, the vector $\boldsymbol{x}^T = (x_1, ..., x_n)$

It is the optimization variable of the problem problem

The function $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$

It is the objective function.

Furthermore

The function $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$

They are the (inequality) constraint functions, and the constants $b_1, ..., b_m$ are the limits.

We want to find a vector

A vector x^* is called optimal, or a solution of the problem.

If it has the smallest objective value among all vectors that satisfy the constraints

for any z with $f_1(z) \leq b_1, ..., f_m(z) \leq b_m$

$$f_0(z) \geq f_i(x^*)$$

Furthermore

The function $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$

They are the (inequality) constraint functions, and the constants $b_1, ..., b_m$ are the limits.

We want to find a vector

A vector $\boldsymbol{x}^*$ is called optimal, or a solution of the problem.

If it has the smallest objective value among all vectors that satisfy the constraints

for any z with $f_1(z) \leq b_1, ..., f_m(z) \leq b_m$

$$f_0(z) \geq f_i(\boldsymbol{x}^*)$$

Furthermore

The function $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$

They are the (inequality) constraint functions, and the constants $b_1, ..., b_m$ are the limits.

We want to find a vector

A vector $\boldsymbol{x}^*$ is called optimal, or a solution of the problem.

If it has the smallest objective value among all vectors that satisfy the constraints

for any z with $f_1(z) \leq b_1, ..., f_m(z) \leq b_m$

$$f_0(z) \geq f_i(\boldsymbol{x}^*)$$

Outline

We have several

An Optimization Problem is called linear program

If the objective and constraint functions $f_0, ..., f_m$ are linear:

$$f_i(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) = \alpha f_i(\boldsymbol{x}) + \beta f_i(\boldsymbol{y})$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$.

If the optimization problem is not linear

It is called a nonlinear program.

We have several

An Optimization Problem is called linear program

If the objective and constraint functions $f_0, ..., f_m$ are linear:

$$f_i\left(\alpha \boldsymbol{x} + \beta \boldsymbol{y}\right) = \alpha f_i\left(\boldsymbol{x}\right) + \beta f_i\left(\boldsymbol{y}\right)$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$.

If the optimization problem is not linear

It is called a nonlinear program.

In our case, we go half way

We have the following type of functions

A convex optimization problem is one in which the objective and constraint functions $f_0, ..., f_m$ are convex:

$$f_i(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) \leq \alpha f_i(\boldsymbol{x}) + \beta f_i(\boldsymbol{y})$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$.

Here, we have constraints

$$\alpha + \beta = 1$$
$$\alpha \geq 0$$
$$\beta \geq 0$$

In our case, we go half way

We have the following type of functions

A convex optimization problem is one in which the objective and constraint functions $f_0, ..., f_m$ are convex:

$$f_i\left(\alpha \boldsymbol{x} + \beta \boldsymbol{y}\right) \leq \alpha f_i\left(\boldsymbol{x}\right) + \beta f_i\left(\boldsymbol{y}\right)$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$.

Here, we have constraints

$$\begin{aligned} \alpha + \beta &= 1 \\ \alpha &\geq 0 \\ \beta &\geq 0 \end{aligned}$$

Outline

Example

In portfolio optimization

We seek the best way to invest some capital in a set of n assets.

Vector representation of Assets

The variable x_i represents the investment in the i^{th} asset:

$$x^T = (x_1, x_2, x_3, ..., x_n)$$

Where the constraints might represent a limit on the budget

A limit on the total amount to be invested

Example

In portfolio optimization

We seek the best way to invest some capital in a set of n assets.

Vector representation of Assets

The variable x_i represents the investment in the i^{th} asset:

$$\boldsymbol{x}^T = (x_1, x_2, x_3, ..., x_n)$$

Where the constraints might represent a limit on the budget

A limit on the total amount to be invested

Example

In portfolio optimization

We seek the best way to invest some capital in a set of n assets.

Vector representation of Assets

The variable x_i represents the investment in the i^{th} asset:

$$\boldsymbol{x}^T = (x_1, x_2, x_3, ..., x_n)$$

Where the constraints might represent a limit on the budget

A limit on the total amount to be invested

For Example

We have the following

The efficient frontier is the curve that shows all efficient portfolios in a risk-return framework:

1. An efficient portfolio is defined as the portfolio that maximizes the expected return for a given amount of risk (standard deviation).
2. The portfolio that minimizes the risk subject to a given expected return.

For Example

We have the following

The efficient frontier is the curve that shows all efficient portfolios in a risk-return framework:

1. An efficient portfolio is defined as the portfolio that maximizes the expected return for a given amount of risk (standard deviation).
2. The portfolio that minimizes the risk subject to a given expected return.

For Example

We have the following

The efficient frontier is the curve that shows all efficient portfolios in a risk-return framework:

1. An efficient portfolio is defined as the portfolio that maximizes the expected return for a given amount of risk (standard deviation).
2. The portfolio that minimizes the risk subject to a given expected return.

For Example

We have the following

The efficient frontier is the curve that shows all efficient portfolios in a risk-return framework:

1. An efficient portfolio is defined as the portfolio that maximizes the expected return for a given amount of risk (standard deviation).
2. The portfolio that minimizes the risk subject to a given expected return.

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- r vector rate of return on assets
- μ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- r vector rate of return on assets
- μ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- r vector rate of return on assets
- μ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- r vector rate of return on assets
- μ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- r vector rate of return on assets
- μ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- $\boldsymbol{r}$ vector rate of return on assets
- $\boldsymbol{\mu}$ vector expected rate of return on assets
- $\boldsymbol{\theta}$ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns $\boldsymbol{r}$

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- $\boldsymbol{r}$ vector rate of return on assets
- $\boldsymbol{\mu}$ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- $\boldsymbol{r}$ vector rate of return on assets
- $\boldsymbol{\mu}$ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns r

We need a little bit of notation

We have

- C_0 capital that can be invested
- C_{end} capital at the end of the period
- R_p total portfolio return
- μ_p expected portfolio return
- σ_p^2 variance of portfolio return
- $\boldsymbol{r}$ vector rate of return on assets
- $\boldsymbol{\mu}$ vector expected rate of return on assets
- θ vector amount invested at each asset
- $\Sigma = [\sigma_{ij}]$ matrix of covariances of returns $\boldsymbol{r}$

Then, we have

The following equalities

- $C_{end} = C_0 + R_p$
- $R_p = \boldsymbol{r}^T \theta$
- $\mu_p = \mu^T \theta$
- $\sigma_p^2 = \theta^2 \Sigma \theta$

Cost Function

We have

$$
\begin{aligned}
var\left(C_{end}\right) &= var\left(C_0 + R_p\right)\\
&= var\left(R_p\right)\\
&= var\left(r^T\theta\right)\\
&= \theta^T \Sigma \theta
\end{aligned}
$$

Cost Function

We have

$$
\begin{aligned}
var\left(C_{end}\right) &= var\left(C_0 + R_p\right)\\
&= var\left(R_p\right)\\
&= var\left(r^T\theta\right)\\
&= \theta^T \Sigma \theta
\end{aligned}
$$

Cost Function

We have

$$
\begin{aligned}
var\left(C_{end}\right) &= var\left(C_0 + R_p\right) \\
&= var\left(R_p\right) \\
&= var\left(r^T \theta\right) \\
&= \theta^T \Sigma \theta
\end{aligned}
$$

Cost Function

We have

$$\begin{aligned} var\left(C_{end}\right) &= var\left(C_0 + R_p\right) \\ &= var\left(R_p\right) \\ &= var\left(r^T\theta\right) \\ &= \theta^T\Sigma\theta \end{aligned}$$

Under the constraints

First, the expected return must be fixed, because we are minimizing the risk given this return

$$\mu^T \theta = \mu_p$$

The second constraint is that we can only invest the capital we have

$$1^T \theta = C_0$$

Under the constraints

First, the expected return must be fixed, because we are minimizing the risk given this return

$$\mu^T \theta = \mu_p$$

The second constraint is that we can only invest the capital we have

$$1^T \theta = C_0$$

Using a little bit of linear algebra

We have that

$$A = \begin{bmatrix} \mu & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} \mu_p \\ C_0 \end{bmatrix}$$

We have that we can rewrite our problem as

$$\min \left\{ \theta^T \Sigma \theta | A^T \theta = B \right\}$$

Using a little bit of linear algebra

We have that

$$A = \begin{bmatrix} \mu & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} \mu_p \\ C_0 \end{bmatrix}$$

We have that we can rewrite our problem as

$$\min \left\{ \theta^T \Sigma \theta | A^T \theta = B \right\}$$

Outline

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

- Solving various classes of optimization problems.
- Analyzing their properties.
- Developing good software implementations.

- Particular forms of the objective function.
- Particular forms of the constraint functions.
- How many variables and constraints there are.

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems.
- Analyzing their properties.
- Developing good software implementations.

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

Firstly

We have that

A solution method for a class of optimization problems is an algorithm that computes a solution of the problem.

Since the late 1940s

A large effort has gone into developing algorithms for:

- Solving various classes of optimization problems,
- Analyzing their properties,
- Developing good software implementations.

Our ability to solve the optimization problems

It varies considerably, depending on factors such as

- Particular forms of the objective function
- Particular forms of the constraint functions
- How many variables and constraints there are

In particular

Special structure

A problem is sparse if each constraint function depends on only a small number of the variables

Something Notable

Even when the objective and constraint functions are smooth

- The optimization is surprisingly hard to solve.

However, for a few problem

We have effective algorithms that can reliably solve even large problems (Thousand of variables and function)

In particular

Special structure

A problem is sparse if each constraint function depends on only a small number of the variables

Something Notable

Even when the objective and constraint functions are smooth

- The optimization is surprisingly hard to solve.

However, for a few problem

We have effective algorithms that can reliably solve even large problems (Thousand of variables and function)

In particular

Special structure

A problem is sparse if each constraint function depends on only a small number of the variables

Something Notable

Even when the objective and constraint functions are smooth

- The optimization is surprisingly hard to solve.

However, for a few problem

We have effective algorithms that can reliably solve even large problems (Thousand of variables and function)

Outline

Introduction

Observation

we describe two very widely known and used special sub-classes of convex optimization:

- Least-Squares Problems
- Linear Programming

Least-Squares Error (LSE)

A least-squares problem is an optimization problem with no constraints

$$\min f_0(\boldsymbol{x}) = \sum_{i=1}^{k} \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i\right)^2 = \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

Remember the Solution

$$\boldsymbol{x} = \left(A^T A\right)^{-1} A^T b$$

The least-squares problem can be solved in a time approximately proportional

$$O\left(n^2 k\right)$$

Least-Squares Error (LSE)

A least-squares problem is an optimization problem with no constraints

$$\min f_0(\boldsymbol{x}) = \sum_{i=1}^{k} \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i\right)^2 = \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

Remember the Solution

$$\boldsymbol{x} = \left(A^T A\right)^{-1} A^T \boldsymbol{b}$$

The least-squares problem can be solved in a time approximately proportional

$$O\left(n^2 k\right)$$

Least-Squares Error (LSE)

A least-squares problem is an optimization problem with no constraints

$$\min f_0\left(\boldsymbol{x}\right) = \sum_{i=1}^{k} \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i\right)^2 = \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

Remember the Solution

$$\boldsymbol{x} = \left(A^T A\right)^{-1} A^T \boldsymbol{b}$$

The least-squares problem can be solved in a time approximately proportional

$$O\left(n^2 k\right)$$

Something Notable

In many cases we can solve even larger least-squares problems

Exploiting a special property of the problem

What if A is sparse?

It has far fewer than kn nonzero entries.

It is possible

To accelerate the solution of the LSE Problem

Something Notable

In many cases we can solve even larger least-squares problems

Exploiting a special property of the problem

What if A is sparse?

It has far fewer than kn nonzero entries.

It is possible

To accelerate the solution of the LSE Problem

Something Notable

In many cases we can solve even larger least-squares problems

Exploiting a special property of the problem

What if A is sparse?

It has far fewer than kn nonzero entries.

It is possible

To accelerate the solution of the LSE Problem

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Examples

- Computational vision (Poggio, Torre, & Koch, 1985; Bertero, Poggio, & Torre, 1988),
- System identification (Akaike, 1974; Johansen, 1997),
- Nonlinear dynamic reconstruction (Haykin, 1999),
- Density estimation (Vapnik, 1998a)
- etc

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Examples

- Computational vision (Poggio, Torre, & Koch, 1985; Bertero, Poggio, & Torre, 1988),
- System identification (Akaike, 1974; Johansen, 1997),
- Nonlinear dynamic reconstruction (Haykin, 1999),
- Density estimation (Vapnik, 1998a)
- etc

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Examples

- Computational vision (Poggio, Torre, & Koch, 1985; Bertero, Poggio, & Torre, 1988),
- System identification (Akaike, 1974; Johansen, 1997),
- Nonlinear dynamic reconstruction (Haykin, 1999),
- Density estimation (Vapnik, 1998a)
- etc

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Examples

- Computational vision (Poggio, Torre, & Koch, 1985; Bertero, Poggio, & Torre, 1988),
- System identification (Akaike, 1974; Johansen, 1997),
- Nonlinear dynamic reconstruction (Haykin, 1999),
- Density estimation (Vapnik, 1998a)
- etc

Regularization

Observation

Most of the inverse problems posed in science and engineering areas are ill posed.

- Basically if you have $Ax = b$ how do you find x?

Examples

- Computational vision (Poggio, Torre, & Koch, 1985; Bertero, Poggio, & Torre, 1988),
- System identification (Akaike, 1974; Johansen, 1997),
- Nonlinear dynamic reconstruction (Haykin, 1999),
- Density estimation (Vapnik, 1998a)
- etc

The house example

Imagine the following data set

Now assume that we use LSE

For the fitting

$$\frac{1}{2}\sum_{i=1}^{N}\left(h_{\boldsymbol{w}}\left(x_i\right)-y_i\right)^2$$

We can then run one of our machine to see what minimize better the previous equation

Question: Did you notice that I did not impose any structure to $h_w(x)$?

Now assume that we use LSE

For the fitting

$$\frac{1}{2}\sum_{i=1}^{N}\left(h_{\boldsymbol{w}}\left(x_i\right)-y_i\right)^2$$

We can then run one of our machine to see what minimize better the previous equation

Question: Did you notice that I did not impose any structure to $h_{\boldsymbol{w}}\left(x\right)$?

Then, First fitting

What about using $h_1(x) = w_0 + w_1 x + w_2 x^2$?

Second fitting

What about using $h_2(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$?

Therefore, we have a problem

We get weird over-fitting effects!!!

What do we do? What about minimizing the influence of w_3, w_4, w_5?

How do we do that?

$$\min_{w} \frac{1}{2} \sum_{i=1}^{N} \left(h_w \left(x_i \right) - y_i \right)^2$$

What about integrating those values to the cost function? Ideas

Therefore, we have a problem

We get weird over-fitting effects!!!

What do we do? What about minimizing the influence of w_3, w_4, w_5?

How do we do that?

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}} \left(x_i \right) - y_i \right)^2$$

What about integrating those values to the cost function? Ideas

We have

Regularization intuition is as follow

Small values for parameters $w_0, w_1, w_2, ..., w_n$

It implies

1. "Simpler" function
2. Less prone to overfitting

We have

Regularization intuition is as follow

Small values for parameters $w_0, w_1, w_2, ..., w_n$

It implies

1. "Simpler" function
2. Less prone to overfitting

We can do the previous idea for the other parameters

We can do the same for the other parameters

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \sum_{i=1}^{d} \lambda_i w_i^2 \quad (1)$$

However handling such many parameters can be so difficult

Combinatorial problem in reality!!!

We can do the previous idea for the other parameters

We can do the same for the other parameters

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \sum_{i=1}^{d} \lambda_i w_i^2 \tag{1}$$

However handling such many parameters can be so difficult

Combinatorial problem in reality!!!

Better, we can

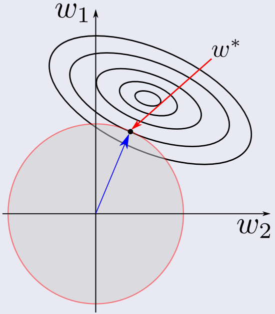
We better use the following

$$\min_{\boldsymbol{w}} \frac{1}{2} \sum_{i=1}^{N} \left(h_{\boldsymbol{w}}\left(x_i\right) - y_i\right)^2 + \lambda \sum_{i=1}^{d} w_i^2 \tag{2}$$

Graphically

Geometrically Equivalent to

$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda \sum_{i=1}^{d+1} w_i^2$



What about Thousands of Features?

There is a technique for that

Least Absolute Shrinkage and Selection Operator (LASSO) invented by Robert Tibshirani that uses $L_1 = \sum_{i=1}^{d} |w_i|$.

The Least Squared Error takes the form of

$$\sum_{i=1}^{N} \left(y_i - x^T w \right)^2 + \sum_{i=1}^{d} |w_i| \quad (3)$$

However

You have other regularizations as $L_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$

What about Thousands of Features?

There is a technique for that

Least Absolute Shrinkage and Selection Operator (LASSO) invented by Robert Tibshirani that uses $L_1 = \sum_{i=1}^{d} |w_i|$.

The Least Squared Error takes the form of

$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w}\right)^2 + \sum_{i=1}^{d} |w_i| \tag{3}$$

However

You have other regularizations as $L_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$

What about Thousands of Features?

There is a technique for that

Least Absolute Shrinkage and Selection Operator (LASSO) invented by Robert Tibshirani that uses $L_1 = \sum_{i=1}^{d} |w_i|$.

The Least Squared Error takes the form of

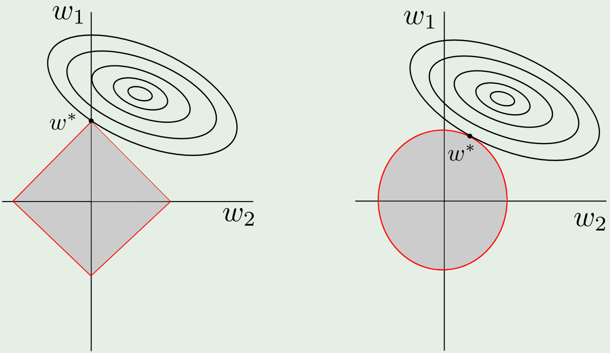
$$\sum_{i=1}^{N} \left(y_i - \boldsymbol{x}^T \boldsymbol{w} \right)^2 + \sum_{i=1}^{d} |w_i| \tag{3}$$

However

You have other regularizations as $L_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$

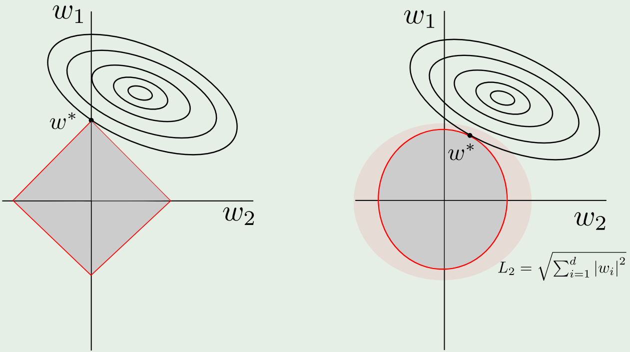
Graphically

The first area correspond to the L_1 regularization and the second one?



Graphically

Yes the circle defined as $L_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$



Outline

Line Segments

Define a segment between two points

Suppose $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ are two points in $\mathbb{R}^n$. Points of the form

$$y = \lambda \boldsymbol{x}_1 + (1 - \lambda)\, \boldsymbol{x}_2$$

where $\lambda \in \mathbb{R}$ form a line passing through $\boldsymbol{x}_1, \boldsymbol{x}_2$.

In the case $\lambda \in (0, 1)$

Then, we have a line segment between x_1 and x_2.

Line Segments

Define a segment between two points

Suppose $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ are two points in $\mathbb{R}^n$. Points of the form

$$y = \lambda \boldsymbol{x}_1 + (1 - \lambda)\, \boldsymbol{x}_2$$

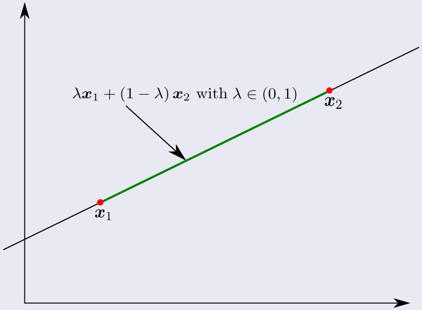
where $\lambda \in \mathbb{R}$ form a line passing through $\boldsymbol{x}_1, \boldsymbol{x}_2$.

In the case $\lambda \in (0, 1)$

Then, we have a line segment between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

Example

We have in $\mathbb{R}^2$



Convex sets

Definition

A set C is convex if the line segment between any two points in C lies in C, i.e. if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and any $\lambda \in [0,1]$

$$\lambda \boldsymbol{x}_1 + (1-\lambda)\,\boldsymbol{x}_2 \in C$$

Additionally

We call a point of the form

$$\lambda_1 \boldsymbol{x}_1 + \ldots + \lambda_n \boldsymbol{x}_n$$

a convex combination of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with $\sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_i \geq 0$

Denoted

$$conv\ C = \{\lambda_1 \boldsymbol{x}_1 + \ldots + \lambda_n \boldsymbol{x}_n | \boldsymbol{x}_i \in C$$
$$\lambda_i \in C \text{ and } \textstyle\sum_{i=1}^{n} \lambda_i = 1\}$$

Convex sets

Definition

A set C is convex if the line segment between any two points in C lies in C, i.e. if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and any $\lambda \in [0,1]$

$$\lambda \boldsymbol{x}_1 + (1-\lambda)\,\boldsymbol{x}_2 \in C$$

Additionally

We call a point of the form

$$\lambda_1 \boldsymbol{x}_1 + ... + \lambda_n \boldsymbol{x}_n$$

a convex combination of points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ with $\sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_i \geq 0$

Denoted

$$conv\ C = \{\lambda_1 \boldsymbol{x}_1 + ... + \lambda_n \boldsymbol{x}_n | \boldsymbol{x}_i \in C$$

$$\lambda_i \in C \text{ and } \sum_{i=1}^{n} \lambda_i = 1\}$$

Convex sets

Definition

A set C is convex if the line segment between any two points in C lies in C, i.e. if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and any $\lambda \in [0,1]$

$$\lambda \boldsymbol{x}_1 + (1-\lambda)\,\boldsymbol{x}_2 \in C$$

Additionally

We call a point of the form

$$\lambda_1 \boldsymbol{x}_1 + ... + \lambda_n \boldsymbol{x}_n$$

a convex combination of points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ with $\sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_i \geq 0$

Denoted

$$conv\ C = \{\lambda_1 \boldsymbol{x}_1 + ... + \lambda_n \boldsymbol{x}_n | \boldsymbol{x}_i \in C$$
$$\lambda_i \in C \text{ and } \sum_{i=1}^{n} \lambda_i{=}1\}$$

It is more

Something Notable

It can be shown that a set is convex if and only if it contains every convex combination of its points.

Property

If B is any convex set that contains C, then $conv\ C \subseteq B$.

It is more

Something Notable

It can be shown that a set is convex if and only if it contains every convex combination of its points.

Property

If B is any convex set that contains C, then $conv\ C \subseteq B$.

Outline

Convexity

Intersection

Convexity is preserved under intersection:

- if S_1 and S_2 are convex, then $S_1 \cap S_2$ is convex.

Affine Transformation

If C is a convex set, $C \subseteq \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, then

$$AC + b = \{Ax + b | x \in C\} \subseteq \mathbb{R}^m$$

Convexity

Intersection

Convexity is preserved under intersection:

- if S_1 and S_2 are convex, then $S_1 \cap S_2$ is convex.

Affine Transformation

If C is a convex set, $C \subseteq \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, then

$$AC + b = \{A\boldsymbol{x} + b | \boldsymbol{x} \in C\} \subseteq \mathbb{R}^m$$

Further

Translation and Scaling

$$C + b, \alpha C$$

Set sum

$$C_1 + C_2 = \{c_1 + c_2 | c_1 \in C_1, c_2 \in C_2\}$$

Further

Translation and Scaling

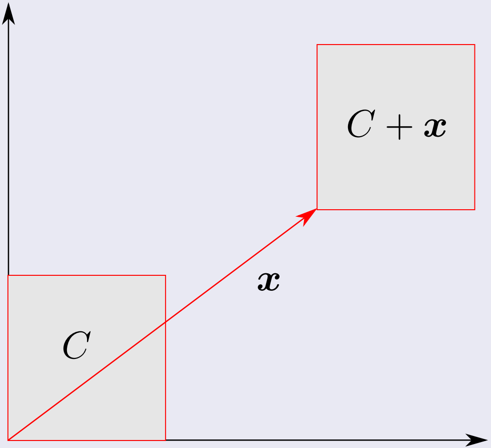
$$C + b, \alpha C$$

Set sum

$$C_1 + C_2 = \{c_1 + c_2 | c_1 \in C_1, c_2 \in C_2\}$$

For Example

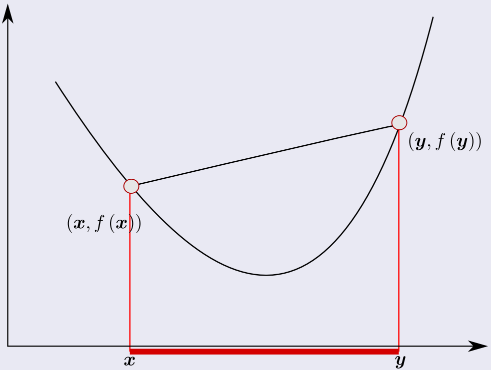
We have a convex set under translation



Outline

We have that

Intuition



Definition

Convex function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $dom(f)$ is a convex set and if$\forall \boldsymbol{x}, \boldsymbol{y} \in dom(f), \forall \theta \in [0,1]$, we have:

$$f\left(\theta \boldsymbol{x} + (1-\theta)\,\boldsymbol{y}\right) \leq \theta f\left(\boldsymbol{x}\right) + (1-\theta)\, f\left(\boldsymbol{y}\right)$$

Something Notable

The epigraph of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the set of points:

$$epi\left(f\right) = \left\{ \left(\boldsymbol{x}, t\right) | \boldsymbol{x} \in dom\left(f\right), t \geq f\left(\boldsymbol{x}\right) \right\}$$

Definition

Convex function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $dom(f)$ is a convex set and if$\forall \boldsymbol{x}, \boldsymbol{y} \in dom(f), \forall \theta \in [0, 1]$, we have:

$$f\left(\theta \boldsymbol{x} + (1 - \theta)\, \boldsymbol{y}\right) \leq \theta f\left(\boldsymbol{x}\right) + (1 - \theta)\, f\left(\boldsymbol{y}\right)$$

Something Notable

The epigraph of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the set of points:

$$epi\left(f\right) = \left\{\left(\boldsymbol{x}, t\right) | \boldsymbol{x} \in dom\left(f\right), t \geq f\left(\boldsymbol{x}\right)\right\}$$

Further

Theorem

The function f is convex if and only if the set $epi\,(f)$ is convex

Proof

Quite simple...

Further

Theorem

The function f is convex if and only if the set $epi\left(f\right)$ is convex

Proof

Quite simple...

Zeroth Order Property

Theorem

f is convex if and only if $\forall \boldsymbol{x} \in dom\left(f\right)$, $\forall \boldsymbol{u}$, the function

$$g\left(t\right) = f\left(\boldsymbol{x} + t\boldsymbol{u}\right)$$

is convex when restricted to the domain $\left\{t | \boldsymbol{x} + t\boldsymbol{u} \in dom\left(f\right)\right\}$

Proof

Look at the board

Zeroth Order Property

Theorem

f is convex if and only if $\forall \boldsymbol{x} \in dom\,(f)$, $\forall \boldsymbol{u}$, the function

$$g\,(t) = f\,(\boldsymbol{x} + t\boldsymbol{u})$$

is convex when restricted to the domain $\{t | \boldsymbol{x} + t\boldsymbol{u} \in dom\,(f)\}$

Proof

Look at the board

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

1. Taking any point x on the $dom(f)$
2. Taking a vertical slice through the point x
3. The resulting plane intersects the domain of f on a line
4. Generating a new 2D function $g(t)$

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point x on the $dom(f)$
3. Taking a vertical slice through the point x
4. The resulting plane intersects the domain of f on a line, $x + tu$
5. Generating a new 2D function $g(t)$

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point x on the $dom\left(f\right)$
3. Taking a vertical slice through the point x
4. The resulting plane intersects the domain of f on a line, $x + tv$
5. Generating a new 2D function $g\left(t\right)$

How does this look like?

Look at the board

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point x on the $dom\,(f)$
3. Taking a vertical slice through the point x
4. The resulting plane intersects the domain of f on a line, $x + tu$
5. Generating a new 2D function $g\,(t)$

How does this look like?

Look at the board

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point $\boldsymbol{x}$ on the $dom\,(f)$
3. Taking a vertical slice through the point $\boldsymbol{x}$
4. The resulting plane intersects the domain of f on a line, $\boldsymbol{x} + t\boldsymbol{u}$
5. Generating a new 2D function $g\,(t)$

How does this look like?

Look at the board

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point $\boldsymbol{x}$ on the $dom\,(f)$
3. Taking a vertical slice through the point $\boldsymbol{x}$
4. The resulting plane intersects the domain of f on a line, $\boldsymbol{x} + t\boldsymbol{u}$
5. Generating a new 2D function $g\,(t)$

How does this look like?

Look at the board

Remark

This property is useful

Checking the convexity of a multivariate function reduces to check the convexity of a univariate function.

Example

1. A 3D convex cup-shaped function
2. Taking any point $\boldsymbol{x}$ on the $dom\,(f)$
3. Taking a vertical slice through the point $\boldsymbol{x}$
4. The resulting plane intersects the domain of f on a line, $\boldsymbol{x} + t\boldsymbol{u}$
5. Generating a new 2D function $g\,(t)$

How does this look like?

Look at the board

Outline

Outline

Suppose f is differentiable

Theorem

Then f is convex if and only if $dom\left(f\right)$ is convex and

$$f\left(\boldsymbol{y}\right) \geq f\left(\boldsymbol{x}\right) + \nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{y} - \boldsymbol{x}\right)$$

holds for all $\boldsymbol{x}$, $\boldsymbol{y} \in dom\left(f\right)$.

- we can derive global information (global underestimator of f)

Suppose f is differentiable

Theorem

Then f is convex if and only if $dom\left(f\right)$ is convex and

$$f\left(\boldsymbol{y}\right) \geq f\left(\boldsymbol{x}\right) + \nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{y} - \boldsymbol{x}\right)$$

holds for all $\boldsymbol{x}$, $\boldsymbol{y} \in dom\left(f\right)$.

Then

- The inequality shows that from local information about a convex function (derivative and value at a point).
- we can derive global information (global underestimator of it).

Suppose f is differentiable

Theorem

Then f is convex if and only if $dom\left(f\right)$ is convex and

$$f\left(\boldsymbol{y}\right) \geq f\left(\boldsymbol{x}\right) + \nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{y} - \boldsymbol{x}\right)$$

holds for all $\boldsymbol{x}$, $\boldsymbol{y} \in dom\left(f\right)$.

Then

- The inequality shows that from local information about a convex function (derivative and value at a point).
- we can derive global information (global underestimator of it).

Minimizing a Convex Function

We have the following situation

- Let $f : \mathbb{R}^n \to \mathbb{R}$ and consider the problem to minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in S$.
- A point $x \in S$ is called a feasible solution to the problem.
- If $f(x) \geq f(\bar{x})$ for each $x \in S$, $\bar{x}$ is called an optimal solution.

Minimizing a Convex Function

We have the following situation

- Let $f : \mathbb{R}^n \to \mathbb{R}$ and consider the problem to minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in S$.
- A point $\boldsymbol{x} \in S$ is called a feasible solution to the problem.
- If $f(\boldsymbol{x}) \geq f(\bar{\boldsymbol{x}})$ for each $\boldsymbol{x} \in S$, $\bar{\boldsymbol{x}}$ is called an optimal solution.

Minimizing a Convex Function

We have the following situation

- Let $f : \mathbb{R}^n \to \mathbb{R}$ and consider the problem to minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in S$.
- A point $\boldsymbol{x} \in S$ is called a feasible solution to the problem.
- If $f(\boldsymbol{x}) \geq f(\bar{\boldsymbol{x}})$ for each $\boldsymbol{x} \in S$, $\bar{\boldsymbol{x}}$ is called an optimal solution.

Specifically

if $\nabla f(\boldsymbol{x}) = 0$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) = f(\boldsymbol{x})$$

Therefore

x is a global minimizer of the function f.

Specifically

if $\nabla f(\boldsymbol{x}) = 0$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) = f(\boldsymbol{x})$$

Therefore

$\boldsymbol{x}$ is a global minimizer of the function f.

Proof

Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex

$$g\left(t\right) = f\left(t\boldsymbol{y} + \left(1 - x\right)\boldsymbol{x}\right)$$

Then, this function is convex and

$$g\left(t\right) = \nabla f\left(t\boldsymbol{y} + \left(1 - t\right)\boldsymbol{x}\right)^T\left(\boldsymbol{y} - \boldsymbol{x}\right)$$

We have the following

$$g\left(\alpha t_1 + \left(1 - \alpha\right) t_2\right) = g\left(t_2 + \alpha\left(t_1 - t_2\right)\right) \leq \alpha g\left(t_1\right) + \left(1 - \alpha\right) g\left(t_2\right)$$

Proof

Let $f:\mathbb{R}^n \to \mathbb{R}$ be convex

$$g(t) = f(t\boldsymbol{y} + (1-x)\boldsymbol{x})$$

Then, this function is convex and

$$g(t) = \nabla f(t\boldsymbol{y} + (1-t)\boldsymbol{x})^T(\boldsymbol{y}-\boldsymbol{x})$$

We have the following

$$g(\alpha t_1 + (1-\alpha)t_2) = g(t_2 + \alpha(t_1 - t_2)) \leq \alpha g(t_1) + (1-\alpha)g(t_2)$$

Proof

Let $f:\mathbb{R}^n \to \mathbb{R}$ be convex

$$g\left(t\right)=f\left(t\boldsymbol{y}+\left(1-x\right)\boldsymbol{x}\right)$$

Then, this function is convex and

$$g\left(t\right)=\nabla f\left(t\boldsymbol{y}+\left(1-t\right)\boldsymbol{x}\right)^{T}\left(\boldsymbol{y}-\boldsymbol{x}\right)$$

We have the following

$$g\left(\alpha t_1+\left(1-\alpha\right)t_2\right)=g\left(t_2+\alpha\left(t_1-t_2\right)\right)\leq \alpha g\left(t_1\right)+\left(1-\alpha\right)g\left(t_2\right)$$

Rearranging Terms

We have

$$g(t_1) \geq g(t_2) + \frac{g(t_2 + \alpha(t_1 - t_2)) - g(t_2)}{\alpha}$$

Making $\alpha \longrightarrow 0$

$$g(t_1) \geq g(t_2) + g'(t_2)(t_1 - t_2)$$

Then $g(1) \geq g(0) + g'(0)$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Rearranging Terms

We have

$$g\left(t_1\right) \geq g\left(t_2\right) + \frac{g\left(t_2 + \alpha\left(t_1 - t_2\right)\right) - g\left(t_2\right)}{\alpha}$$

Making $\alpha \longrightarrow 0$

$$g\left(t_1\right) \geq g\left(t_2\right) + g'\left(t_2\right)\left(t_1 - t_2\right)$$

Then $g\left(1\right) \geq g\left(0\right) + g'\left(0\right)$

$$f\left(\boldsymbol{y}\right) \geq f\left(\boldsymbol{x}\right) + \nabla f\left(\boldsymbol{x}\right)^T\left(\boldsymbol{y} - \boldsymbol{x}\right)$$

Rearranging Terms

We have

$$g(t_1) \geq g(t_2) + \frac{g(t_2 + \alpha(t_1 - t_2)) - g(t_2)}{\alpha}$$

Making $\alpha \longrightarrow 0$

$$g(t_1) \geq g(t_2) + g'(t_2)(t_1 - t_2)$$

Then $g(1) \geq g(0) + g'(0)$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x})$$

The other part of the proof

I leave you the part

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) \implies f \text{ is convex}$$

Outline

Here, we assume

If f is twice differentiable

The Hessian exist!!!

Definition

Given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, then the Jacobian of the derivatives

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}$$

is called the Hessian Matrix H

I.e.

$$Hf = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Here, we assume

If f is twice differentiable

The Hessian exist!!!

Definition

Given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, then the Jacobian of the derivatives

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}$$

is called the Hessian Matrix H

I.e.

$$Hf = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Here, we assume

If f is twice differentiable

The Hessian exist!!!

Definition

Given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, then the Jacobian of the derivatives

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}$$

is called the Hessian Matrix H

I.e.

$$Hf = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Then, we need also

Definition

A real matrix A is positive if

$$x^T Ax > 0$$

Theorem

Let f be a twice differentiable function on an open domain $dom(f)$. The f is convex if and only if $dom(f)$ is convex and its Hessian is positive semidefinite:

$$x^T Hx \geq 0$$

Then, we need also

Definition

A real matrix A is positive if

$$x^T A x > 0$$

Theorem

Let f be a twice differentiable function on an open domain $dom(f)$. The f is convex if and only if $dom(f)$ is convex and its Hessian is positive semidefinite:

$$x^T H x \geq 0$$

For example

In the case for functions on $\mathbb{R}$

We have the simple condition $f''(x)$

- Which means that the first derivative $f'(x)$ is non-decreasing.

Outline

Then

Just as for convex sets

We consider standard operations which preserve function convexity.

Basically

We can say if a function is convex then it is constructed from simpler convex functions

Basically

A way to test for convexity

Then

Just as for convex sets

We consider standard operations which preserve function convexity.

Basically

We can say if a function is convex then it is constructed from simpler convex functions

Basically

A way to test for convexity

Then

Just as for convex sets

We consider standard operations which preserve function convexity.

Basically

We can say if a function is convex then it is constructed from simpler convex functions

Basically

A way to test for convexity

We have

Non-negative weighted sum

$$\forall \alpha_i \geq 0, \; \sum_{i=1}^{k} \alpha_i f_i$$

Composition with affine mapping

$$f(Ax+b)$$

Composition with monotone convex

$g(f(x))$ is convex for f convex, g convex and non-decreasing.

We have

Non-negative weighted sum

$$\forall \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i f_i$$

Composition with affine mapping

$$f\left(A\boldsymbol{x} + b\right)$$

Composition with monotone convex

$g\left(f\left(x\right)\right)$ is convex for f convex, g convex and non-decreasing.

We have

Non-negative weighted sum

$$\forall \alpha_i \geq 0, \ \sum_{i=1}^{k} \alpha_i f_i$$

Composition with affine mapping

$$f\left(A\boldsymbol{x} + b\right)$$

Composition with monotone convex

$g\left(f\left(x\right)\right)$ is convex for f convex, g convex and non-decreasing.

How

We have

$$h''(x) = [g(f(x))]''$$

Then

$$h''(x) = [g'(f(x))\, f'(x)]' = (g''(f(x)))\,[f'(x)]^2 + g'(f(x))\, f''(x) \geq 0$$

Because

- If f, g are convex than $g''(f(x))$ and $f''(x)$ are both ≥ 0
- if g is non-decreasing, $g'(f(x))$ is also ≥ 0

How

We have

$$h''(x) = [g(f(x))]''$$

Then

$$h''(x) = [g'(f(x))\, f'(x)]' = (g''(f(x)))\,[f'(x)]^2 + g'(f(x))\, f''(x) \geq 0$$

Because

- If f , g are convex than $g''(f(x))$ and $f''(x)$ are both ≥ 0
- if g is non-decreasing, $g'(f(x))$ is also ≥ 0

How

We have

$$h''(x) = [g(f(x))]''$$

Then

$$h''(x) = [g'(f(x))\,f'(x)]' = (g''(f(x)))\,[f'(x)]^2 + g'(f(x))\,f''(x) \geq 0$$

Because

- If f , g are convex than $g''(f(x))$ and $f''(x)$ are both ≥ 0
- if g is non-decreasing, $g'(f(x))$ is also ≥ 0

Outline

Introduction

Optimization Searches

Sometimes referred to as iterative improvement or local search.

Introduction

Optimization Searches

Sometimes referred to as iterative improvement or local search.

There are different techniques

1. Gradient Descent
2. Hillclimbing
3. Random Restart Hillclimbing
4. Simulated Annealing

Introduction

Optimization Searches

Sometimes referred to as iterative improvement or local search.

There are different techniques

1. Gradient Descent
2. Hillclimbing
3. Random Restart Hillclimbing
4. Simulated Annealing

Introduction

Optimization Searches

Sometimes referred to as iterative improvement or local search.

There are different techniques

1. Gradient Descent
2. Hillclimbing
3. Random Restart Hillclimbing
4. Simulated Annealing

Introduction

Optimization Searches

Sometimes referred to as iterative improvement or local search.

There are different techniques

1. Gradient Descent
2. Hillclimbing
3. Random Restart Hillclimbing
4. Simulated Annealing

All of them have something in common

Local Search

Algorithm that explores the search space of possible solutions in sequential fashion, moving from a current state to a "nearby" one.

All of them have something in common

Local Search

Algorithm that explores the search space of possible solutions in sequential fashion, moving from a current state to a "nearby" one.

Why is this important?

- Set of configurations may be too large to be enumerated explicitly.
- Might there not be a polynomial time algorithm for finding the maximum of the problem efficiently.
  - Thus local improvements can be a solution to the problem

All of them have something in common

Local Search

Algorithm that explores the search space of possible solutions in sequential fashion, moving from a current state to a "nearby" one.

Why is this important?

- Set of configurations may be too large to be enumerated explicitly.
- Might there not be a polynomial time algorithm for finding the maximum of the problem efficiently.
  - Thus local improvements can be a solution to the problem

All of them have something in common

Local Search

Algorithm that explores the search space of possible solutions in sequential fashion, moving from a current state to a "nearby" one.

Why is this important?

- Set of configurations may be too large to be enumerated explicitly.
- Might there not be a polynomial time algorithm for finding the maximum of the problem efficiently.
  - Thus local improvements can be a solution to the problem

Local Search Facts

What is interesting about Local Search

- It keeps track of single current state
- It move only to neighboring states

Local Search Facts

What is interesting about Local Search

- It keeps track of single current state
- It move only to neighboring states

Advantages

- It uses very little memory
- It can often find reasonable solutions in large or infinite (continuous) state spaces

Local Search Facts

What is interesting about Local Search

- It keeps track of single current state
- It move only to neighboring states

Advantages

- It uses very little memory
- It can often find reasonable solutions in large or infinite (continuous) state spaces

Local Search Facts

What is interesting about Local Search

- It keeps track of single current state
- It move only to neighboring states

Advantages

- It uses very little memory
- It can often find reasonable solutions in large or infinite (continuous) state spaces

Outline

What happen if you have the following

What if you have a cost function with the following characteristics

- It is parametrically defined.
- It is smooth.

What happen if you have the following

What if you have a cost function with the following characteristics

- It is parametrically defined.
- It is smooth.

We can use the following technique

Gradient Descent

What happen if you have the following

What if you have a cost function with the following characteristics

- It is parametrically defined.
- It is smooth.

We can use the following technique

Gradient Descent

Example

Consider the following hypothetical problem

1. x =sales price of Intel's newest chip (in $1000's of dollars)
2. $f(x)$ =profit per chip when it costs $1000.00 dollars

Example

Consider the following hypothetical problem

1. x =sales price of Intel's newest chip (in \$1000's of dollars)
2. $f(x)$ =profit per chip when it costs \$1000.00 dollars

Assume that Intel's marketing research team has found that the profit per chip (as a function of x) is

$$f(x) = x^2 - x^3$$

Example

Consider the following hypothetical problem

1. x =sales price of Intel's newest chip (in \$1000's of dollars)
2. $f(x)$ =profit per chip when it costs \$1000.00 dollars

Assume that Intel's marketing research team has found that the profit per chip (as a function of x) is

$$f(x) = x^2 - x^3$$

Assume

we must have x non-negative and no greater than one in percentage.

Example

Consider the following hypothetical problem

1. x =sales price of Intel's newest chip (in $1000's of dollars)
2. $f(x)$ =profit per chip when it costs $1000.00 dollars

Assume that Intel's marketing research team has found that the profit per chip (as a function of x) is

$$f(x) = x^2 - x^3$$

Assume

we must have x non-negative and no greater than one in percentage.

Thus

Maximization

Objective function is profit $f(x)$ that needs to be maximized.

Thus

Solution to the optimization problem will be the optimum chip sales price.

Thus

Maximization

Objective function is profit $f(x)$ that needs to be maximized.

Thus

Solution to the optimization problem will be the optimum chip sales price.

Outline

Important Notes about Optimization Problems

What we want

We are interested in knowing those points $\boldsymbol{x} \in D \subseteq \mathbb{R}^n$ such that $f(\boldsymbol{x}_0) \leq f(\boldsymbol{x})$ of $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x})$

Or

A minimum or a maximum point x_0.

The process of finding x_0

It is a search process using certain properties of the function.

Important Notes about Optimization Problems

What we want

We are interested in knowing those points $\boldsymbol{x} \in D \subseteq \mathbb{R}^n$ such that $f(\boldsymbol{x}_0) \leq f(\boldsymbol{x})$ of $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x})$

Or

A minimum or a maximum point $\boldsymbol{x}_0$.

The process of finding x_0

It is a search process using certain properties of the function.

Important Notes about Optimization Problems

What we want

We are interested in knowing those points $\boldsymbol{x} \in D \subseteq \mathbb{R}^n$ such that $f(\boldsymbol{x}_0) \leq f(\boldsymbol{x})$ of $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x})$

Or

A minimum or a maximum point $\boldsymbol{x}_0$.

The process of finding $\boldsymbol{x}_0$

It is a search process using certain properties of the function.

Thus

Local vs Global Minimum/Maximum

- Local minimum/maximum is the minimum/maximum in a neighborhood $L \subset D$.
- Global minimum/maximum is the lowest value of f for all $x \in D$
  - it is usually much harder to find.

Thus

Local vs Global Minimum/Maximum

- Local minimum/maximum is the minimum/maximum in a neighborhood $L \subset D$.
- Global minimum/maximum is the lowest value of f for all $x \in D$
  - it is usually much harder to find.

Examples of minimums

Thus

Local vs Global Minimum/Maximum

- Local minimum/maximum is the minimum/maximum in a neighborhood $L \subset D$.
- Global minimum/maximum is the lowest value of f for all $x \in D$
  - **it is usually much harder to find.**

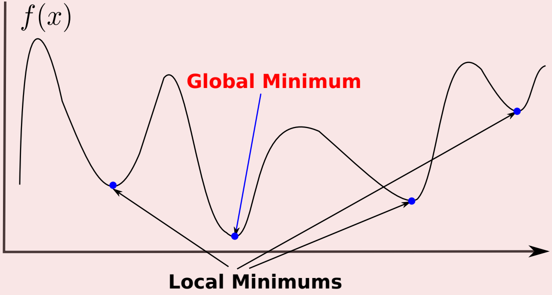
Examples of minimums

Thus

Local vs Global Minimum/Maximum

- Local minimum/maximum is the minimum/maximum in a neighborhood $L \subset D$.
- Global minimum/maximum is the lowest value of f for all $x \in D$
  - **it is usually much harder to find.**

Examples of minimums



Furthermore

Something Notable

Optimization is a very difficult problem in general.

- Especially when x is high dimensional, unless f is simple (e.g. linear) and known analytically.

Furthermore

Something Notable

Optimization is a very difficult problem in general.

- Especially when x is high dimensional, unless f is simple (e.g. linear) and known analytically.

We have this classification

1. Analytical methods - They work fine when f can be handled in an analytical way.
2. Numerical methods - Here, we use inherent properties of the function like the rate of change of the function.

Furthermore

Something Notable

Optimization is a very difficult problem in general.

- Especially when x is high dimensional, unless f is simple (e.g. linear) and known analytically.

We have this classification

1. Analytical methods - They work fine when f can be handled in an analytical way.
2. Numerical methods - Here, we use inherent properties of the function like the rate of change of the function.

In our case

We will look at the Gradient Descent Method!!!

Furthermore

Something Notable

Optimization is a very difficult problem in general.

- Especially when x is high dimensional, unless f is simple (e.g. linear) and known analytically.

We have this classification

1. Analytical methods - They work fine when f can be handled in an analytical way.
2. Numerical methods - Here, we use inherent properties of the function like the rate of change of the function.

In our case

We will look at the Gradient Descent Method!!!

Furthermore

Something Notable

Optimization is a very difficult problem in general.

- Especially when x is high dimensional, unless f is simple (e.g. linear) and known analytically.

We have this classification

1. Analytical methods - They work fine when f can be handled in an analytical way.
2. Numerical methods - Here, we use inherent properties of the function like the rate of change of the function.

In our case

We will look at the Gradient Descent Method!!!

Analytical Method: Differentiating

Assume f is known analytically and twice differentiable

The critical points of f, i.e. the points of potential maximum or minimum, can be found using the equation:

$$\frac{df}{dx} = 0 \quad (4)$$

Analytical Method: Differentiating

Assume f is known analytically and twice differentiable

The critical points of f, i.e. the points of potential maximum or minimum, can be found using the equation:

$$\frac{df}{dx} = 0 \tag{4}$$

For example

$$\frac{df(x)}{dx} = \frac{d\left[x^2 - x^3\right]}{dx} = 2x - 3x^2 = 0$$

Analytical Method: Differentiating

Assume f is known analytically and twice differentiable

The critical points of f, i.e. the points of potential maximum or minimum, can be found using the equation:

$$\frac{df}{dx} = 0 \tag{4}$$

For example

$$\frac{df(x)}{dx} = \frac{d\left[x^2 - x^3\right]}{dx} = 2x - 3x^2 = 0$$

Finding the roots $x_1, x_2, \ldots, x_k$

$$x = \frac{2}{3}$$

Analytical Method: Differentiating

Assume f is known analytically and twice differentiable

The critical points of f, i.e. the points of potential maximum or minimum, can be found using the equation:

$$\frac{df}{dx} = 0 \tag{4}$$

For example

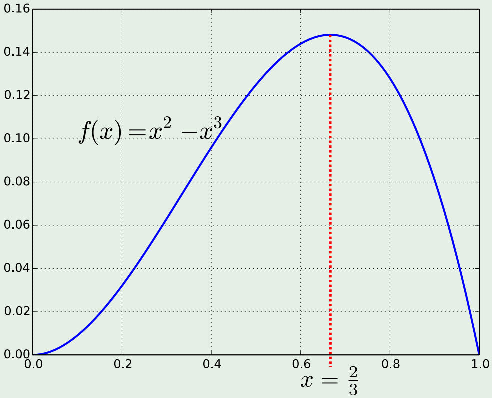
$$\frac{df(x)}{dx} = \frac{d\left[x^2 - x^3\right]}{dx} = 2x - 3x^2 = 0$$

Finding the roots $x_1, x_2, ..., x_k$

$$x = \frac{2}{3}$$

Example

We have the following



Do we have a Maximum or a Minimum

Second Derivative Test

The sign of the second derivative tells if each of those points is a maximum or a minimum:

1. If $\frac{d^2 f(x_i)}{dx^2} > 0$ for $x = x_i$ then x_i is a minimum.
2. If $\frac{d^2 f(x_i)}{dx^2} < 0$ for $x = x_i$ then x_i is a maximum.

Do we have a Maximum or a Minimum

Second Derivative Test

The sign of the second derivative tells if each of those points is a maximum or a minimum:

1. If $\frac{d^2 f(x_i)}{dx^2} > 0$ for $x = x_i$then x_i is a minimum.
2. If $\frac{d^2 f(x_i)}{dx^2} < 0$ for $x = x_i$then x_i is a maximum.

Do we have a Maximum or a Minimum

Second Derivative Test

The sign of the second derivative tells if each of those points is a maximum or a minimum:

1. If $\frac{d^2 f(x_i)}{dx^2} > 0$ for $x = x_i$then x_i is a minimum.
2. If $\frac{d^2 f(x_i)}{dx^2} < 0$ for $x = x_i$then x_i is a maximum.

Do we have a Maximum or a Minimum

Second Derivative Test

The sign of the second derivative tells if each of those points is a maximum or a minimum:

1. If $\frac{d^2 f(x_i)}{dx^2} > 0$ for $x = x_i$then x_i is a minimum.
2. If $\frac{d^2 f(x_i)}{dx^2} < 0$ for $x = x_i$then x_i is a maximum.

Example

In our case

$$\frac{d^2 f(x)}{dx^2} = 2 - 6x$$

Then

$$\frac{d^2 f\left(\frac{2}{3}\right)}{dx^2} = 2 - 6 \times \frac{2}{3} = 2 - 4 = -2$$

Maximum Profit for the $1000.00 dollar Chip

$$\$667.00$$

Example

In our case

$$\frac{d^2 f(x)}{dx^2} = 2 - 6x$$

Then

$$\frac{d^2 f\left(\frac{2}{3}\right)}{dx^2} = 2 - 6 \times \frac{2}{3} = 2 - 4 = -2$$

Maximum Profit for the $1000.00 dollar Chip

$$\$667.00$$

Example

In our case

$$\frac{d^2 f(x)}{dx^2} = 2 - 6x$$

Then

$$\frac{d^2 f\left(\frac{2}{3}\right)}{dx^2} = 2 - 6 \times \frac{2}{3} = 2 - 4 = -2$$

Maximum Profit for the $1000.00 dollar Chip

$$\$667.00$$

What if $\frac{d^2 f(x_i)}{dx^2} = 0$?

Question

If the second derivative is 0 in a critical point x_i, then x_i may or may not be a minimum or a maximum of f. **WHY?**

We have for $x^3 - 3x^2 + x - 2$

With derivative

$$\frac{d^2 f(x)}{dx^2} = 6x - 6$$

What if $\frac{d^2 f(x_i)}{dx^2} = 0$?

Question

If the second derivative is 0 in a critical point x_i, then x_i may or may not be a minimum or a maximum of f. **WHY?**

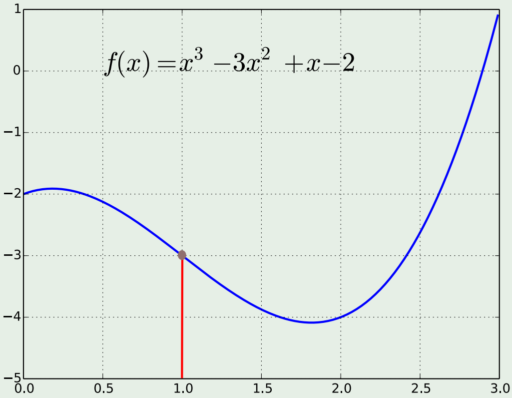
We have for $x^3 - 3x^2 + x - 2$

With derivative

$$\frac{d^2 f(x)}{dx^2} = 6x - 6$$

Actually a point where $\frac{d^2 f(x_i)}{dx^2} = 0$

We have a change in the “curvature $\approxeq \frac{d^2 f(x)}{dx^2}$”



Properties of Differentiating

Generalization

To move to higher dimensional functions, we will require to take partial derivatives!!!

Solving

A system of equations!!!

Remark

For a bounded D the only possible points of maximum/minimum are critical or boundary ones, so, in principle, we can find the global extremum.

Properties of Differentiating

Generalization

To move to higher dimensional functions, we will require to take partial derivatives!!!

Solving

A system of equations!!!

Remark

For a bounded D the only possible points of maximum/minimum are critical or boundary ones, so, in principle, we can find the global extremum.

Properties of Differentiating

Generalization

To move to higher dimensional functions, we will require to take partial derivatives!!!

Solving

A system of equations!!!

Remark

For a bounded D the only possible points of maximum/minimum are critical or boundary ones, so, in principle, we can find the global extremum.

Problems

A lot of them

- Potential problems include transcendent equations, not solvable analytically.
- High cost of finding derivatives, especially in high dimensions (e.g. for neural networks)

Thus

Partial Solution of the problems comes from a numerical technique called the gradient descent

Problems

A lot of them

- Potential problems include transcendent equations, not solvable analytically.
- High cost of finding derivatives, especially in high dimensions (e.g. for neural networks)

Thus

Partial Solution of the problems comes from a numerical technique called the gradient descent

Outline

Numerical Method: Gradient Descent

Imagine the following

- f is a smooth objective function.
- Now you have a x_0 state and you need to find the next one.

Numerical Method: Gradient Descent

Imagine the following

- f is a smooth objective function.
- Now you have a x_0 state and you need to find the next one.

Something Notable

We want to find x in the neighborhood D of x_0 such that

$$f(x) < f(x_0)$$

Numerical Method: Gradient Descent

Imagine the following

- f is a smooth objective function.
- Now you have a $\boldsymbol{x}_0$ state and you need to find the next one.

Something Notable

We want to find $\boldsymbol{x}$ in the neighborhood D of $\boldsymbol{x}_0$ such that

$$f(\boldsymbol{x}) < f(\boldsymbol{x}_0)$$

Taylor's Expansion

Using the first order Taylor's expansion around point $x \in \mathbb{R}^n$ for $f : \mathbb{R}^n \to \mathbb{R}$

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + O\left(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2\right)$$

Note:

- Actually the Taylor's expansions are polynomial approximation to the function!!!
- $\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, ..., \frac{\partial f(x)}{\partial x_n}\right]^T$ with $x = (x_1, x_2, ..., x_n)^T$

Taylor's Expansion

Using the first order Taylor's expansion around point $\boldsymbol{x} \in \mathbb{R}^n$ for $f : \mathbb{R}^n \to \mathbb{R}$

$$f\left(\boldsymbol{x}\right) = f\left(\boldsymbol{x}_0\right) + \nabla f\left(\boldsymbol{x}_0\right)^T \cdot \left(\boldsymbol{x} - \boldsymbol{x}_0\right) + O\left(\left\|\boldsymbol{x} - \boldsymbol{x}_0\right\|^2\right)$$

Note:

- Actually the Taylor's expansions are polynomial approximation to the function!!!
- $\nabla f\left(\boldsymbol{x}\right) = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, ..., \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T$ with $\boldsymbol{x} = \left(x_1, x_2, ..., x_n\right)^T$

If we can find a neighborhood D small enough

We can discard the terms of the second and higher orders because the linear approximation is enough!!!

Taylor's Expansion

Using the first order Taylor's expansion around point $\boldsymbol{x} \in \mathbb{R}^n$ for $f : \mathbb{R}^n \to \mathbb{R}$

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + O\left(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2\right)$$

Note:

- Actually the Taylor's expansions are polynomial approximation to the function!!!
- $\nabla f(\boldsymbol{x}) = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, ..., \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T$ with $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$

If we can find a neighborhood D small enough

We can discard the terms of the second and higher orders because the linear approximation is enough!!!

Taylor's Expansion

Using the first order Taylor's expansion around point $\boldsymbol{x} \in \mathbb{R}^n$ for $f : \mathbb{R}^n \to \mathbb{R}$

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + O\left(\|\boldsymbol{x} - \boldsymbol{x}_0\|^2\right)$$

Note:

- Actually the Taylor's expansions are polynomial approximation to the function!!!
- $\nabla f(\boldsymbol{x}) = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, ..., \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T$ with $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T$

If we can find a neighborhood D small enough

We can discard the terms of the second and higher orders because the linear approximation is enough!!!

How do we do this?

Simple

$$\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{u}$$

where x_0 and u are vectors and h is a constant.

How do we do this?

Simple

$$\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{u}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{u}$ are vectors and h is a constant.

Then we get

$$f(\boldsymbol{x}_0 + h\boldsymbol{u}) - f(\boldsymbol{x}_0) = h\nabla f(\boldsymbol{x}_0)^T \cdot \boldsymbol{u} + h^2 O(1)$$

How do we do this?

Simple

$$\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{u}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{u}$ are vectors and h is a constant.

Then we get

$$f(\boldsymbol{x}_0 + h\boldsymbol{u}) - f(\boldsymbol{x}_0) = h\nabla f(\boldsymbol{x}_0)^T \cdot \boldsymbol{u} + h^2 O(1)$$

We make h^2 term insignificant by shrinking h

Thus, if we want to decrease $f(\boldsymbol{x}_0 + h\boldsymbol{u}) - f(\boldsymbol{x}_0) < 0$ the fastest, enforcing $f(\boldsymbol{x}_0 + h\boldsymbol{u}) < f(\boldsymbol{x}_0)$:

$$f(\boldsymbol{x}_0 + h\boldsymbol{u}) - f(\boldsymbol{x}_0) \approx h\nabla f(\boldsymbol{x}_0)^T \cdot \boldsymbol{u}$$

How do we do this?

Simple

$$\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{u}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{u}$ are vectors and h is a constant.

Then we get

$$f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) = h\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u} + h^2 O\left(1\right)$$

We make h^2 term insignificant by shrinking h

Thus, if we want to decrease $f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) < 0$ the fastest, enforcing $f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) < f\left(\boldsymbol{x}_0\right)$:

$$f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) \approx h\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u}$$

How do we do this?

Simple

$$\boldsymbol{x} = \boldsymbol{x}_0 + h\boldsymbol{u}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{u}$ are vectors and h is a constant.

Then we get

$$f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) = h\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u} + h^2 O\left(1\right)$$

We make h^2 term insignificant by shrinking h

Thus, if we want to decrease $f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) < 0$ the fastest, enforcing $f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) < f\left(\boldsymbol{x}_0\right)$:

$$f\left(\boldsymbol{x}_0 + h\boldsymbol{u}\right) - f\left(\boldsymbol{x}_0\right) \approx h\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u}$$

Then

We minimize

$$\nabla f(\boldsymbol{x}_0)^T \cdot \boldsymbol{u}$$

Thus, the unit vector that minimize

In order to obtain the largest difference

$$\boldsymbol{u} = -\frac{\nabla f(\boldsymbol{x}_0)}{\|\nabla f(\boldsymbol{x}_0)\|}$$

Then

$$\nabla f(\boldsymbol{x}_0)^T \times -\frac{\nabla f(\boldsymbol{x}_0)}{\|\nabla f(\boldsymbol{x}_0)\|} = -\|\nabla f(\boldsymbol{x}_0)\| < 0$$

Then

We minimize

$$\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u}$$

Thus, the unit vector that minimize

In order to obtain the largest difference

$$\boldsymbol{u} = -\frac{\nabla f\left(\boldsymbol{x}_0\right)}{\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\|}$$

Then

$$\nabla f\left(\boldsymbol{x}_0\right)^T \times -\frac{\nabla f\left(\boldsymbol{x}_0\right)}{\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\|} = -\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\| < 0$$

Then

We minimize

$$\nabla f\left(\boldsymbol{x}_0\right)^T \cdot \boldsymbol{u}$$

Thus, the unit vector that minimize

In order to obtain the largest difference

$$\boldsymbol{u} = -\frac{\nabla f\left(\boldsymbol{x}_0\right)}{\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\|}$$

Then

$$\nabla f\left(\boldsymbol{x}_0\right)^T \times -\frac{\nabla f\left(\boldsymbol{x}_0\right)}{\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\|} = -\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\| < 0$$

Therefore

We have that

$$
\begin{aligned}
\boldsymbol{x} &= \boldsymbol{x}_0 + h\boldsymbol{u} \\
&= \boldsymbol{x}_0 - h\frac{\nabla f(\boldsymbol{x}_0)}{\|\nabla f(\boldsymbol{x}_0)\|} \\
&= \boldsymbol{x}_0 - h'\nabla f(\boldsymbol{x}_0)
\end{aligned}
$$

With $h' = \frac{h}{\|\nabla f(\boldsymbol{x}_0)\|}$

Therefore

We have that

$$
\begin{aligned}
\boldsymbol{x} &= \boldsymbol{x}_0 + h\boldsymbol{u} \\
&= \boldsymbol{x}_0 - h\frac{\nabla f(\boldsymbol{x}_0)}{\|\nabla f(\boldsymbol{x}_0)\|} \\
&= \boldsymbol{x}_0 - h'\nabla f(\boldsymbol{x}_0)
\end{aligned}
$$

With $h' = \frac{h}{\|\nabla f(\boldsymbol{x}_0)\|}$

Therefore

We have that

$$
\begin{aligned}
\boldsymbol{x} &= \boldsymbol{x}_0 + h\boldsymbol{u} \\
&= \boldsymbol{x}_0 - h\frac{\nabla f\left(\boldsymbol{x}_0\right)}{\left\|\nabla f\left(\boldsymbol{x}_0\right)\right\|} \\
&= \boldsymbol{x}_0 - h'\nabla f\left(\boldsymbol{x}_0\right)
\end{aligned}
$$

With $h' = \frac{h}{\|\nabla f(\boldsymbol{x}_0)\|}$

Outline

Gradient Descent

In the method of Gradient descent, we have a cost function $J(\boldsymbol{w})$ where

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta \nabla J(\boldsymbol{w}(\boldsymbol{n}))$$

How, we prove that $J(\boldsymbol{w}(n+1)) < J(\boldsymbol{w}(n))$?

We use the first-order Taylor series expansion around $\boldsymbol{w}(n)$

$$J(\boldsymbol{w}(n+1)) \approx J(\boldsymbol{w}(n)) + \nabla J^T(\boldsymbol{w}(\boldsymbol{n}))\Delta\boldsymbol{w}(n) \quad (5)$$

Remark: This is quite true when the step size is quite small!!! In addition, $\Delta\boldsymbol{w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n)$

Gradient Descent

In the method of Gradient descent, we have a cost function $J(\boldsymbol{w})$ where

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \eta \nabla J(\boldsymbol{w}(\boldsymbol{n}))$$

How, we prove that $J(\boldsymbol{w}(n+1)) < J(\boldsymbol{w}(n))$?

We use the first-order Taylor series expansion around $\boldsymbol{w}(n)$

$$J(\boldsymbol{w}(n+1)) \approx J(\boldsymbol{w}(n)) + \nabla J^T(\boldsymbol{w}(\boldsymbol{n}))\Delta\boldsymbol{w}(n) \quad (5)$$

Remark: This is quite true when the step size is quite small!!! In addition, $\Delta\boldsymbol{w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n)$

Why? Look at the case in $\mathbb{R}$

The equation of the tangent line to the curve $y = J(w(n))$

$$L(w(n)) = J'(w(n))[w(n+1) - w(n)] + J(w(n)) \tag{6}$$

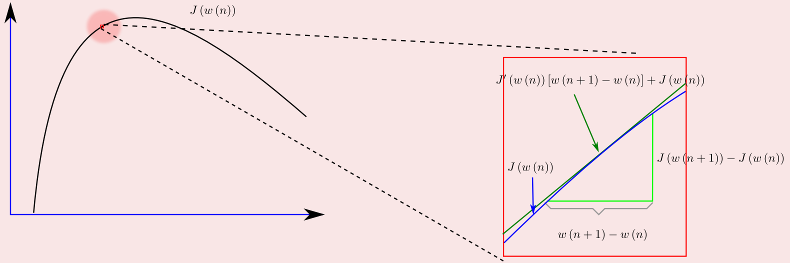
Example

Why? Look at the case in $\mathbb{R}$

The equation of the tangent line to the curve $y = J(w(n))$

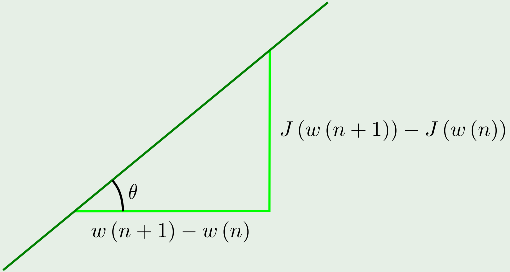
$$L(w(n)) = J'(w(n))[w(n+1) - w(n)] + J(w(n)) \tag{6}$$

Example



Thus, we have that in $\mathbb{R}$

Remember Something quite Classic



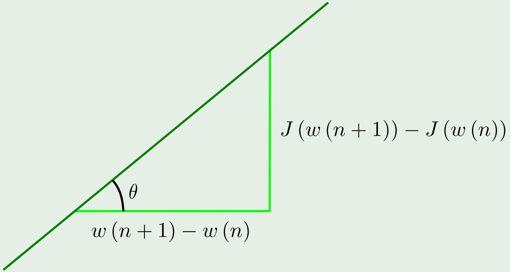
$$\tan\theta = \frac{J(w(n+1)) - J(w(n))}{w(n+1) - w(n)}$$

$$\tan\theta\,(w(n+1) - w(n)) = J(w(n+1)) - J(w(n))$$

$$J'(w(n))\,(w(n+1) - w(n)) = J(w(n+1)) - J(w(n))$$

Thus, we have that in $\mathbb{R}$

Remember Something quite Classic



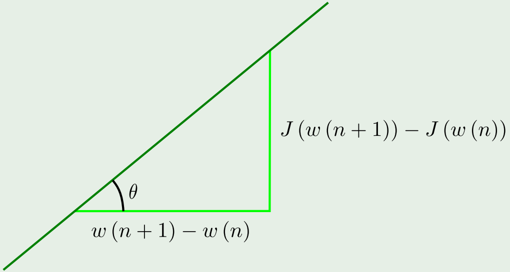
$$\tan\theta = \frac{J\left(w\left(n+1\right)\right)-J\left(w\left(n\right)\right)}{w\left(n+1\right)-w\left(n\right)}$$

$$\tan\theta\left(w\left(n+1\right)-w\left(n\right)\right) = J\left(w\left(n+1\right)\right)-J\left(w\left(n\right)\right)$$

$$J'\left(w\left(n\right)\right)\left(w\left(n+1\right)-w\left(n\right)\right) = J\left(w\left(n+1\right)\right)-J\left(w\left(n\right)\right)$$

Thus, we have that in $\mathbb{R}$

Remember Something quite Classic



$$\tan\theta = \frac{J(w(n+1)) - J(w(n))}{w(n+1) - w(n)}$$
$$\tan\theta\,(w(n+1) - w(n)) = J(w(n+1)) - J(w(n))$$
$$J'(w(n))\,(w(n+1) - w(n)) = J(w(n+1)) - J(w(n))$$

Thus, we have that

Using the First Taylor expansion

$$J(w(n)) \approx J(w(n)) + J'(w(n))[w(n+1) - w(n)] \quad (7)$$

Now, for Many Variables

An hyperplane in $\mathbb{R}^n$ is a set of the form

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T \boldsymbol{x} = b \right\} \tag{8}$$

Given $\boldsymbol{x} \in H$ and $\boldsymbol{x}_0 \in H$

$$b = \boldsymbol{a}^T \boldsymbol{x} = \boldsymbol{a}^T \boldsymbol{x}_0$$

Thus, we have that

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T (\boldsymbol{x} - \boldsymbol{x}_0) = 0 \right\}$$

Now, for Many Variables

An hyperplane in $\mathbb{R}^n$ is a set of the form

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T \boldsymbol{x} = b \right\} \quad (8)$$

Given $\boldsymbol{x} \in H$ and $\boldsymbol{x}_0 \in H$

$$b = \boldsymbol{a}^T \boldsymbol{x} = \boldsymbol{a}^T \boldsymbol{x}_0$$

Thus, we have that

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T (\boldsymbol{x} - \boldsymbol{x}_0) = 0 \right\}$$

Now, for Many Variables

An hyperplane in $\mathbb{R}^n$ is a set of the form

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T \boldsymbol{x} = b \right\} \tag{8}$$

Given $\boldsymbol{x} \in H$ and $\boldsymbol{x}_0 \in H$

$$b = \boldsymbol{a}^T \boldsymbol{x} = \boldsymbol{a}^T \boldsymbol{x}_0$$

Thus, we have that

$$H = \left\{ \boldsymbol{x} | \boldsymbol{a}^T \left(\boldsymbol{x} - \boldsymbol{x}_0 \right) = 0 \right\}$$

Thus, we have the following definition

Definition (Differentiability)

Assume that J is defined in a disk D containing $\boldsymbol{w}(n)$. We say that J is differentiable at $\boldsymbol{w}(n)$ if:

1. $\frac{\partial J(\boldsymbol{w}(n))}{\partial w_i}$ exist for all $i = 1, ..., n$.
2. J is locally linear at $\boldsymbol{w}(n)$.

Thus, we have the following definition

Definition (Differentiability)

Assume that J is defined in a disk D containing $\boldsymbol{w}(n)$. We say that J is differentiable at $\boldsymbol{w}(n)$ if:

1. $\frac{\partial J(\boldsymbol{w}(n))}{\partial w_i}$ exist for all $i = 1, ..., n$.
2. J is locally linear at $\boldsymbol{w}(n)$.

Thus, we have the following definition

Definition (Differentiability)

Assume that J is defined in a disk D containing $\boldsymbol{w}(n)$. We say that J is differentiable at $\boldsymbol{w}(n)$ if:

1. $\frac{\partial J(\boldsymbol{w}(n))}{\partial w_i}$ exist for all $i = 1, ..., n$.
2. J is locally linear at $\boldsymbol{w}(n)$.

Thus, given $J\left(\boldsymbol{w}\left(n\right)\right)$

We know that we have the following operator

$$\nabla=\left(\frac{\partial}{\partial w_1},\frac{\partial}{\partial w_2},...,\frac{\partial}{\partial w_m}\right) \tag{9}$$

Thus, given $J(\boldsymbol{w}(n))$

We know that we have the following operator

$$\nabla = \left(\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, ..., \frac{\partial}{\partial w_m}\right) \tag{9}$$

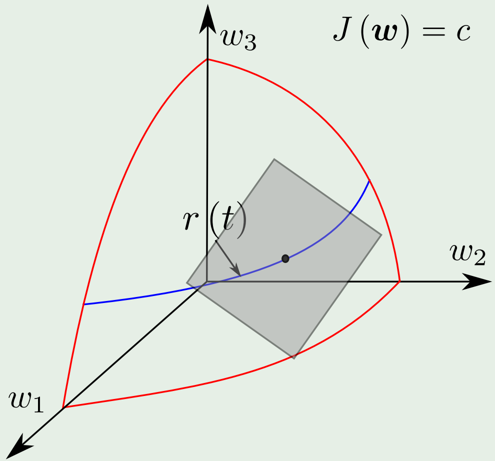
Thus, we have

$$\begin{aligned}\nabla J(\boldsymbol{w}(n)) &= \left(\frac{\partial J(\boldsymbol{w}(n))}{\partial w_1}, \frac{\partial J(\boldsymbol{w}(n))}{\partial w_2}, ..., \frac{\partial J(\boldsymbol{w}(n))}{\partial w_m}\right) \\ &= \sum_{i=1}^{m} \hat{w}_i \frac{\partial J(\boldsymbol{w}(n))}{\partial w_i}\end{aligned}$$

Where: $\hat{w}_i^T = (1, 0, ..., 0) \in \mathbb{R}$

Now

Given a curve function $r(t)$ that lies on the level set $J(\boldsymbol{w}(n)) = c$ (When is in $\mathbb{R}^3$)



Level Set

Definition

$$\{(w_1, w_2, ..., w_m) \in \mathbb{R}^m | J\,(w_1, w_2, ..., w_m) = c\} \quad (10)$$

Remark: In a normal Calculus course we will use x and f instead of $\boldsymbol{w}$ and J.

Where

Any curve has the following parametrization

$$r : [a, b] \to \mathbb{R}^m$$
$$r(t) = (w_1(t), ..., w_m(t))$$

With $r(n+1) = (w_1(n+1), ..., w_m(n+1))$

We can write the parametrized version of it

$$z(t) = J(w_1(t), w_2(t), ..., w_m(t)) = c \quad (11)$$

Differentiating with respect to t and using the chain rule for multiple variables

$$\frac{dz(t)}{dt} = \sum_{i=1}^{m} \frac{\partial J(\boldsymbol{w}(t))}{\partial w_i} \cdot \frac{dw_i(t)}{dt} = 0 \quad (12)$$

Where

Any curve has the following parametrization

$$r : [a, b] \to \mathbb{R}^m$$
$$r(t) = (w_1(t), ..., w_m(t))$$

With $r(n+1) = (w_1(n+1), ..., w_m(n+1))$

We can write the parametrized version of it

$$z(t) = J(w_1(t), w_2(t), ..., w_m(t)) = c \quad (11)$$

Differentiating with respect to t and using the chain rule for multiple variables

$$\frac{dz(t)}{dt} = \sum_{i=1}^{m} \frac{\partial J(\boldsymbol{w}(t))}{\partial w_i} \cdot \frac{dw_i(t)}{dt} = 0 \quad (12)$$

Where

Any curve has the following parametrization

$$r : [a, b] \to \mathbb{R}^m$$
$$r(t) = (w_1(t), ..., w_m(t))$$

With $r(n+1) = (w_1(n+1), ..., w_m(n+1))$

We can write the parametrized version of it

$$z(t) = J(w_1(t), w_2(t), ..., w_m(t)) = c \quad (11)$$

Differentiating with respect to t and using the chain rule for multiple variables

$$\frac{dz(t)}{dt} = \sum_{i=1}^{m} \frac{\partial J(\boldsymbol{w}(t))}{\partial w_i} \cdot \frac{dw_i(t)}{dt} = 0 \quad (12)$$

Note

First

Given $y = f(\boldsymbol{u}) = (f_1(\boldsymbol{u}), ..., f_l(\boldsymbol{u}))$ and $\boldsymbol{u} = g(\boldsymbol{x}) = (g_1(\boldsymbol{x}), ..., g_m(\boldsymbol{x}))$.

We have then that

$$\frac{\partial(f_1, f_2, ..., f_l)}{\partial(x_1, x_2, ..., x_k)} = \frac{\partial(f_1, f_2, ..., f_l)}{\partial(g_1, g_2, ..., g_m)} \cdot \frac{\partial(g_1, g_2, ..., g_m)}{\partial(x_1, x_2, ..., x_k)} \quad (13)$$

Thus

$$\begin{aligned}\frac{\partial(f_1, f_2, ..., f_l)}{\partial x_i} &= \frac{\partial(f_1, f_2, ..., f_l)}{\partial(g_1, g_2, ..., g_m)} \cdot \frac{\partial(g_1, g_2, ..., g_m)}{\partial x_i} \\ &= \sum_{k=1}^{m} \frac{\partial(f_1, f_2, ..., f_l)}{\partial g_k} \frac{\partial g_k}{\partial x_i}\end{aligned}$$

Note

First

Given $y = f(\boldsymbol{u}) = (f_1(\boldsymbol{u}), \ldots, f_l(\boldsymbol{u}))$ and $\boldsymbol{u} = g(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x}))$.

We have then that

$$\frac{\partial (f_1, f_2, \ldots, f_l)}{\partial (x_1, x_2, \ldots, x_k)} = \frac{\partial (f_1, f_2, \ldots, f_l)}{\partial (g_1, g_2, \ldots, g_m)} \cdot \frac{\partial (g_1, g_2, \ldots, g_m)}{\partial (x_1, x_2, \ldots, x_k)} \tag{13}$$

Thus

$$\begin{aligned}\frac{\partial (f_1, f_2, \ldots, f_l)}{\partial x_i} &= \frac{\partial (f_1, f_2, \ldots, f_l)}{\partial (g_1, g_2, \ldots, g_m)} \cdot \frac{\partial (g_1, g_2, \ldots, g_m)}{\partial x_i} \\ &= \sum_{k=1}^{m} \frac{\partial (f_1, f_2, \ldots, f_l)}{\partial g_k} \frac{\partial g_k}{\partial x_i}\end{aligned}$$

Note

First

Given $y = f(\boldsymbol{u}) = (f_1(\boldsymbol{u}), ..., f_l(\boldsymbol{u}))$ and $\boldsymbol{u} = g(\boldsymbol{x}) = (g_1(\boldsymbol{x}), ..., g_m(\boldsymbol{x}))$.

We have then that

$$\frac{\partial (f_1, f_2, ..., f_l)}{\partial (x_1, x_2, ..., x_k)} = \frac{\partial (f_1, f_2, ..., f_l)}{\partial (g_1, g_2, ..., g_m)} \cdot \frac{\partial (g_1, g_2, ..., g_m)}{\partial (x_1, x_2, ..., x_k)} \tag{13}$$

Thus

$$\begin{aligned} \frac{\partial (f_1, f_2, ..., f_l)}{\partial x_i} &= \frac{\partial (f_1, f_2, ..., f_l)}{\partial (g_1, g_2, ..., g_m)} \cdot \frac{\partial (g_1, g_2, ..., g_m)}{\partial x_i} \\ &= \sum_{k=1}^{m} \frac{\partial (f_1, f_2, ..., f_l)}{\partial g_k} \frac{\partial g_k}{\partial x_i} \end{aligned}$$

Thus

Evaluating at $t = n$

$$\sum_{i=1}^{m} \frac{\partial J\left(\boldsymbol{w}\left(n\right)\right)}{\partial w_i} \cdot \frac{dw_i(n)}{dt} = 0$$

We have that

$$\nabla J\left(\boldsymbol{w}\left(n\right)\right) \cdot r'\left(n\right) = 0 \tag{14}$$

This proves that for every level set the gradient is perpendicular to the tangent to any curve that lies on the level set

In particular to the point $\boldsymbol{w}\left(n\right)$.

Thus

Evaluating at $t = n$

$$\sum_{i=1}^{m} \frac{\partial J\left(\boldsymbol{w}\left(n\right)\right)}{\partial w_i} \cdot \frac{dw_i(n)}{dt} = 0$$

We have that

$$\nabla J\left(\boldsymbol{w}\left(n\right)\right) \cdot r'\left(n\right) = 0 \tag{14}$$

This proves that for every level set the gradient is perpendicular to the tangent to any curve that lies on the level set

In particular to the point $w\left(n\right)$.

Thus

Evaluating at $t = n$

$$\sum_{i=1}^{m} \frac{\partial J\left(\boldsymbol{w}\left(n\right)\right)}{\partial w_i} \cdot \frac{dw_i(n)}{dt} = 0$$

We have that

$$\nabla J\left(\boldsymbol{w}\left(n\right)\right) \cdot r'\left(n\right) = 0 \tag{14}$$

This proves that for every level set the gradient is perpendicular to the tangent to any curve that lies on the level set

In particular to the point $\boldsymbol{w}\left(n\right)$.

Now the tangent plane to the surface can be described generally

Thus

$$L\left(\boldsymbol{w}\left(n+1\right)\right) = J\left(\boldsymbol{w}\left(n\right)\right) + \nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\left[\boldsymbol{w}\left(n+1\right) - \boldsymbol{w}\left(n\right)\right] \quad (15)$$

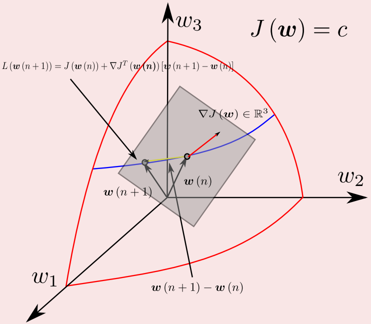
This looks like

Now the tangent plane to the surface can be described generally

Thus

$$L\left(\boldsymbol{w}\left(n+1\right)\right)=J\left(\boldsymbol{w}\left(n\right)\right)+\nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\left[\boldsymbol{w}\left(n+1\right)-\boldsymbol{w}\left(n\right)\right] \qquad (15)$$

This looks like



Proving the fact about the Gradient Descent

We want the following

$$J\left(\boldsymbol{w}\left(n+1\right)\right) < J\left(\boldsymbol{w}\left(n\right)\right)$$

Using the first-order Taylor approximation

$$J\left(\boldsymbol{w}\left(n+1\right)\right) - J\left(\boldsymbol{w}\left(n\right)\right) \approx \nabla J^{T}\left(\boldsymbol{w}\left(n\right)\right)\Delta\boldsymbol{w}\left(n\right)$$

So, we ask the following

$$\Delta\boldsymbol{w}\left(n\right) \approx -\eta\nabla J\left(\boldsymbol{w}\left(n\right)\right) \text{ with } \eta > 0$$

Proving the fact about the Gradient Descent

We want the following

$$J\left(\boldsymbol{w}\left(n+1\right)\right) < J\left(\boldsymbol{w}\left(n\right)\right)$$

Using the first-order Taylor approximation

$$J\left(\boldsymbol{w}\left(n+1\right)\right) - J\left(\boldsymbol{w}\left(n\right)\right) \approx \nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\Delta\boldsymbol{w}\left(n\right)$$

So, we ask the following

$$\Delta\boldsymbol{w}\left(n\right) \approx -\eta\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right) \text{ with } \eta > 0$$

Proving the fact about the Gradient Descent

We want the following

$$J\left(\boldsymbol{w}\left(n+1\right)\right) < J\left(\boldsymbol{w}\left(n\right)\right)$$

Using the first-order Taylor approximation

$$J\left(\boldsymbol{w}\left(n+1\right)\right) - J\left(\boldsymbol{w}\left(n\right)\right) \approx \nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\Delta\boldsymbol{w}\left(n\right)$$

So, we ask the following

$$\Delta\boldsymbol{w}\left(n\right) \approx -\eta\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right) \text{ with } \eta > 0$$

Then

We have that

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)\approx-\eta\nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)=-\eta\left\|\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\right\|^{2}$$

Thus

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)<0$$

Or

$$J\left(\boldsymbol{w}\left(n+1\right)\right)<J\left(\boldsymbol{w}\left(n\right)\right)$$

Then

We have that

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)\approx-\eta\nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)=-\eta\left\|\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\right\|^{2}$$

Thus

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)<0$$

Or

$$J\left(\boldsymbol{w}\left(n+1\right)\right)<J\left(\boldsymbol{w}\left(n\right)\right)$$

Then

We have that

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)\approx-\eta\nabla J^{T}\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)=-\eta\left\|\nabla J\left(\boldsymbol{w}\left(\boldsymbol{n}\right)\right)\right\|^{2}$$

Thus

$$J\left(\boldsymbol{w}\left(n+1\right)\right)-J\left(\boldsymbol{w}\left(n\right)\right)<0$$

Or

$$J\left(\boldsymbol{w}\left(n+1\right)\right)<J\left(\boldsymbol{w}\left(n\right)\right)$$

Outline

Algorithm of Gradient Descent

Initialization

1. Guess an init point x_0
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
   1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Algorithm of Gradient Descent

Initialization

1. Guess an init point $\boldsymbol{x}_0$
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
   1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Algorithm of Gradient Descent

Initialization

1. Guess an init point $\boldsymbol{x}_0$
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
   1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Algorithm of Gradient Descent

Initialization

1. Guess an init point $\boldsymbol{x}_0$
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
    1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Algorithm of Gradient Descent

Initialization

1. Guess an init point $\boldsymbol{x}_0$
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
   1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Algorithm of Gradient Descent

Initialization

1. Guess an init point $\boldsymbol{x}_0$
2. Use a N_{max} iteration count
3. A gradient norm tolerance ϵ_g to know if we have arrived to a critical point.
4. A step tolerance ϵ_x to know if we have done significant progress
5. α_t is known as the step size.
    1. It is chosen to maintain a balance between convergence speed and avoiding divergence.

Finally

Gradient_Descent($\boldsymbol{x}_0, N_{max}, \epsilon_g, \epsilon_t, \alpha_t$)

1. **for** $t = 0, 1, 2, \ldots, N_{max}$
2. $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t)$

Finally

Gradient_Descent$(\boldsymbol{x}_0, N_{max}, \epsilon_g, \epsilon_t, \alpha_t)$

1. **for** $t = 0, 1, 2, ..., N_{max}$
2. $\quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t)$
3. $\quad$ **if** $\|\nabla f(\boldsymbol{x}_{t+1})\| < \epsilon_g$
4. $\quad\quad$ **return “Converged on critical point”**
5. $\quad$ **if** $\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| < \epsilon_t$
6. $\quad\quad$ **return “Converged on an x value”**
7. $\quad$ **if** $f(\boldsymbol{x}_{t+1}) > f(\boldsymbol{x}_t)$
8. $\quad\quad$ **return “Diverging”**
9. **return “Maximum number of iterations reached”**

Finally

Gradient_Descent$(\boldsymbol{x}_0, N_{max}, \epsilon_g, \epsilon_t, \alpha_t)$

1. **for** $t = 0, 1, 2, ..., N_{max}$
2. $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \nabla f(\boldsymbol{x}_t)$
3. **if** $\|\nabla f(\boldsymbol{x}_{t+1})\| < \epsilon_g$
4. **return "Converged on critical point"**
5. **if** $\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| < \epsilon_t$
6. **return "Converged on an x value"**
7. **if** $f(\boldsymbol{x}_{t+1}) > f(\boldsymbol{x}_t)$
8. **return "Diverging"**
9. **return "Maximum number of iterations reached"**

Finally

Gradient_Descent$(\boldsymbol{x}_0, N_{max}, \epsilon_g, \epsilon_t, \alpha_t)$

for $t = 0, 1, 2, ..., N_{max}$

  $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \nabla f\left(\boldsymbol{x}_t\right)$

  if $\|\nabla f\left(\boldsymbol{x}_{t+1}\right)\| < \epsilon_g$

    return “Converged on critical point”

  if $\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| < \epsilon_t$

    return “Converged on an x value”

  if $f\left(\boldsymbol{x}_{t+1}\right) > f\left(\boldsymbol{x}_t\right)$

    return “Diverging”

return “Maximum number of iterations reached”

Finally

Gradient_Descent$(\boldsymbol{x}_0, N_{max}, \epsilon_g, \epsilon_t, \alpha_t)$

```
for t = 0, 1, 2, ..., N_max
        x_{t+1} = x_t − α_t ∇f(x_t)
        if ‖∇f(x_{t+1})‖ < ε_g
                return "Converged on critical point"
        if ‖x_t − x_{t+1}‖ < ε_t
                return "Converged on an x value"
        if f(x_{t+1}) > f(x_t)
                return "Diverging"
return "Maximum number of iterations reached"
```

IMPORTANT

I forgot to mention something

$\nabla f(x)$ give us the direction of the fastest change at x.

IMPORTANT

I forgot to mention something

$\nabla f(x)$ give us the direction of the fastest change at x.

Observations

- Gradient descent can only work if at least we can differentiate the cost function
- Gradient descent gets bottled up in local minima or maxima

IMPORTANT

I forgot to mention something

$\nabla f(x)$ give us the direction of the fastest change at x.

Observations

- Gradient descent can only work if at least we can differentiate the cost function
- Gradient descent gets bottled up in local minima or maxima

Outline

Given that the Canonical Solution has problems

We can develop a more robust algorithm

Using the Gradient Descent Idea

Basically, The Gradient Descent

It uses the change in the surface of the cost function to obtain a direction of improvement.

Given that the Canonical Solution has problems

We can develop a more robust algorithm

Using the Gradient Descent Idea

Basically, The Gradient Descent

It uses the change in the surface of the cost function to obtain a direction of improvement.

Gradient Descent

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)) \quad (16)$$

$\eta(k)$ is a positive scale factor or learning rate!!!

Gradient Descent

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $w(2)$ by moving from $w(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k)) \quad (16)$$

$\eta(k)$ is a positive scale factor or learning rate!!!

Gradient Descent

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k)) \quad (16)$$

$\eta(k)$ is a positive scale factor or learning rate!!!

Gradient Descent

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k)) \tag{16}$$

$\eta(k)$ is a positive scale factor or learning rate!!!

Gradient Descent

The basic procedure is as follow

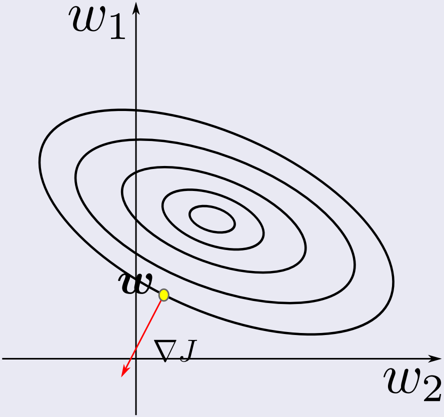
1. Start with a random weight vector $\boldsymbol{w}(1)$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}(1))$.
3. Obtain value $\boldsymbol{w}(2)$ by moving from $\boldsymbol{w}(1)$ in the direction of the steepest descent:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k)) \quad (16)$$

$\eta(k)$ is a positive scale factor or learning rate!!!

Geometrically

We have the following



Outline

For our full regularized equation

We have

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d+1} w_j^2 \tag{17}$$

Then, for each w_j

$$\frac{dJ(\boldsymbol{w})}{dw_j} = -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_j^i\right] + \lambda w_j \tag{18}$$

Therefore

$$\nabla J(\boldsymbol{w}(k)) = \begin{pmatrix} -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_1^i\right] + \lambda w_1 \\ \vdots \\ -\sum_{i=1}^{N}\left[\left(y_i - \sum_{j=1}^{d+1} x_j^i w_j\right) x_{d+1}^i\right] + \lambda w_{d+1} \end{pmatrix}$$

For our full regularized equation

We have

$$J\left(\boldsymbol{w}\right)=\frac{1}{2}\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{d+1}x_j^i w_j\right)^2+\frac{\lambda}{2}\sum_{j=1}^{d+1}w_j^2 \tag{17}$$

Then, for each w_j

$$\frac{dJ\left(\boldsymbol{w}\right)}{dw_j}=-\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^i w_j\right)x_j^i\right]+\lambda w_j \tag{18}$$

Therefore

$$\nabla J\left(w\left(k\right)\right)=\begin{pmatrix}-\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^i w_j\right)x_1^i\right]+\lambda w_1\\ \vdots\\ -\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^i w_j\right)x_{d+1}^i\right]+\lambda w_{d+1}\end{pmatrix}$$

For our full regularized equation

We have

$$J\left(\boldsymbol{w}\right)=\frac{1}{2}\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{d+1}x_j^iw_j\right)^2+\frac{\lambda}{2}\sum_{j=1}^{d+1}w_j^2 \tag{17}$$

Then, for each w_j

$$\frac{dJ\left(\boldsymbol{w}\right)}{dw_j}=-\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^iw_j\right)x_j^i\right]+\lambda w_j \tag{18}$$

Therefore

$$\nabla J\left(\boldsymbol{w}\left(k\right)\right)=\begin{pmatrix} -\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^iw_j\right)x_1^i\right]+\lambda w_1 \\ \vdots \\ -\sum_{i=1}^{N}\left[\left(y_i-\sum_{j=1}^{d+1}x_j^iw_j\right)x_{d+1}^i\right]+\lambda w_{d+1} \end{pmatrix}$$

Outline

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\quad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

- If $\eta(k)$ is too large, correction will overshoot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\quad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k)\nabla J(\boldsymbol{w}(k-1))$$
4. until $\eta(k)\nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\qquad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\qquad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k)\nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k)\nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $\qquad \boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!

Algorithm

Gradient Decent

1. Initialize $\boldsymbol{w}$, criterion θ, $\eta(\cdot)$, $k = 0$
2. do $k = k + 1$
3. $$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \eta(k) \nabla J(\boldsymbol{w}(k-1))$$
4. until $\eta(k) \nabla J(\boldsymbol{w}(k)) < \theta$
5. return $\boldsymbol{w}$

Problem!!! How to choose the learning rate?

- If $\eta(k)$ is too small, convergence is quite slow!!!
- If $\eta(k)$ is too large, correction will overshot and can even diverge!!!