Optimization

Lecture 6: Gradient descent algorithms for unconstrained problems

---

Parin Chaipunya
KMUTT
└Mathematics @ Faculty of Science
└The Joint Graduate School of Energy and Environments

Areas of research:

- Multi-agent optimization: Bilevel programs, Game theory
- Optimization modeling: mainly focused on energy and environmental applications

parin.cha@kmutt.ac.th
parinchaipunya.com
github.com/parinchaipunya

Last update: Januaray 2026

Table of contents

Problem scopes

Consider an unconstrained optimization problem

$$\begin{cases} \min & J(x) \\ \text{s.t.} & x \in \mathcal{X}, \end{cases}$$

where $J : \mathcal{X} \to \mathbb{R}$ and the domain $\mathcal{X} \subset \mathbb{R}^n$ is nonempty and open.

Section 1

Gradient descent algorithm

Subsection 1

Gradient descent schematic

General schematic

To develop an algorithm, we follow the following general schematic.

Algorithm 1 General schematic

Require: Objective J, initial point x^0, tolerance $\varepsilon > 0$
Ensure: Approximate stationary point $\|\nabla J(x^k)\| \leq \varepsilon$
1: $k \leftarrow 0$
2: **while** $\|\nabla J(x^k)\| > \varepsilon$ **do**
3: Choose a direction d^k
4: Choose step size $\gamma_k > 0$
5: $x^{k+1} \leftarrow x^k + \gamma_k d^k$
6: $k \leftarrow k + 1$
7: **end while**
8: **return** x^k

To ensure/enhance convergence, we are required to compute a good *direction* and *step size*.

Gradient descent direction

At each point $x \in \mathcal{X}$, we know that $-\nabla J(x)$ is the direction of **steepest descent**. This fact motivates us to use this direction of descent in Algorithm 1:

$$d^k = -\nabla J(x^k)\,.$$

Section 2

Line search methods

Line search methods

It remains to determine an appropriate step size γ_k. This process of determining γ_k is called **line search**, and it turns out to be the key to obtain the convergence of the algorithm.

The intuition is that
- if γ_k is too small, then the algorithm may have a very slow convergence;
- if γ_k is too large, then the algorithm may overshoot a minimizer or produce divergence and instability.

The common line search methods are:
- **Constant step size method.** Fix $\gamma_k := \bar{\gamma} > 0$ for all k.
- **Exact line search.** Choose γ_k that minimizes $J(x)$ over the ray $\gamma(\gamma) = x^k + \gamma d^k$, that is

$$\gamma_k \in \arg\min_{\gamma \geq 0} J(x^k + \gamma d^k)\,.$$

- **Vanishing step size method.** Choose *a priori* a decreasing positive sequence (γ_k) with $\gamma_k \to 0$.
- **Backtracking line search.** Obtain a sufficient reduction while maintaining speed. (Next slide.)

Backtracking line search

Algorithm 2 Backtracking line search

Require: Objective J, current point x^k, descent direction d^k, parameters $\alpha \in (0,1)$, $\beta \in (0,1)$
Ensure: Sufficient descent $J(x^k + \gamma_k d^k) \leq J(x^k) + \alpha\gamma_k [\nabla J(x^k)]^t d^k$.

1: $\gamma_k \leftarrow 1$
2: **while** $J(x^k + \gamma_k d^k) > J(x^k) + \alpha\gamma_k \nabla J(x^k)^\top d^k$ **do**
3: $\quad \gamma_k \leftarrow \beta\gamma_k$
4: **end while**
5: **return** γ_k

For convenience, we shall write a notation

$$\gamma_k \leftarrow \text{Backtracking}_{\alpha,\beta}(x^k, d^k)$$

as a shorthand for

"γ_k is the output from the backtracking line search with the corresponding inputs."

Backtracking line search

Theorem 1
The backtracking line search (Algorithm 2) terminates in a finite steps.

Gradient descent algorithm with backtracking line search

To conclude, we list in the following the gradient descent algorithm with backtracking line search.

Algorithm 3 Gradient descent algorithm with backtracking line search

Require: Objective J, initial point x^0, tolerance $\varepsilon > 0$, backtracking parameters $\alpha, \beta \in (0, 1)$
Ensure: Approximate stationary point $\|\nabla J(x^k)\| \leq \varepsilon$

$k \leftarrow 0$
while $\|\nabla J(x^k)\| > \varepsilon$ **do**
    Compute the negative gradient direction $d^k = -\nabla J(x^k)$
    Compute the backtracking step size $\gamma_k \leftarrow \text{Backtracking}_{\alpha,\beta}(x^k, d^k)$
    $x^{k+1} \leftarrow x^k + \gamma_k d^k$
    $k \leftarrow k + 1$
end while
return x^k

Accumulation property

Theorem 2

Let (x^k) be a sequence generated by the gradient descent algorithm with backtracking line search (Algorithm 3). If $\bar{x}$ is a limit point of (x^k), then $\nabla J(\bar{x}) = 0$.

Convergence property

Theorem 3

Let J be strongly convex and $\mathcal{D} := S_J(J(w))$ is closed for some $w \in \mathcal{X}$. Then the sequence (x^k) generated by the gradient descent with backtracking line search (Algorithm 3), with initial point $x^0 \in \mathcal{D}$, is convergent to the unique minimizer $\bar{x}$ of J.

Section 3

Toy experiments

Numerical experiments (Unique minimizer, strongly convex)

Example 4

Consider $J : \mathbb{R}^2 \to \mathbb{R}$ defined by $J(x) = x_1^2 + \nu x_2^2$, with $\nu > 0$. Try experimenting with the following line search rules at different values of ν.

- Constant step size with different values of $\bar{\gamma}$.
- Exact line search.
- Vanishing step size method with different sequences (γ_k).
- Back tracking line search with different parameters.

Numerical experiments (Multiple minimizers)

Example 5

Consider $J : \mathbb{R}^4 \to \mathbb{R}$ defined by

$$J(x) = \|Ax - b\|^2 ,$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} .$$

Experiment gradient descent algorithm with different starting points.

Numerical experiments (No minimizer)

Example 6

Consider $J : \mathbb{R} \to \mathbb{R}$ defined by $J(x) = x^3$. Experiment the gradient descent algorithm with different starting points.

Section 4

Applications

Subsection 1

Least squares

Least squares

We have already experimented with a least squares problem from Example 5.

Let us formally recall that a least squares problem is given by

$$\min_{x \in \mathbb{R}^n} \quad J(x) = \|Ax - b\|^2 , \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

We could use the input gradient by the formula $\nabla J(x) = 2A^t(Ax - b)$.

Apart from the least squares problem that solves for a least squares solution of a linear system $Ax = b$, there are actually *many more applications* that fall into the form of (1).

Subsection 2

Linear regression

Linear regression

Linear regression is a problem of fitting a linear hyperplane to a set of observations such that the mean square error (MSE) is minimized.

Let $\left\{(x^d, y^d) \in \mathbb{R}^N \times \mathbb{R}\right\}_{i=1,\ldots,D}$ be a set consisting of D samples (or observations). Here, we develop the following terminology.

- The components of $x^d = (x_1^d, \ldots, x_N^d)$ are called **features** (or **inputs**).
- The vectors x^d's are called **feature vectors** (or **input vectors**).
- The real value y^d's are called **responses** (or **outputs**).

We also have a hypothesis that the outcome y^d's are related via the linear equation

$$y^d \approx w_0 + w_1 x_1^d + \cdots + w_N x_N^d\,, \qquad \forall d = 1, \ldots, D\,,$$

for some **weight vector** $w = (w_0, w_1, \ldots, w_N) \in \mathbb{R}^{N+1}$.

Optimal weight vector

When a weight vector $w = (w_0, w_1, \ldots, w_N) \in \mathbb{R}^{N+1}$ is given, we could evaluate the accuracy of the model by the following **squared error** (or **SE**)

$$J(w) = \sum_{d=1}^{D} \left(w_0 + w_1 x_1^d + \cdots + w_N x_N^d - y^d \right)^2,$$

which will be **minimized** in order to find an **optimal weight vector**.

Vectorization

To efficiently find the optimal weight vector, it is practically useful to do write the above formulation in a matrix-vector form.

Let us write

$$Y = \begin{bmatrix} y^1 \\ \vdots \\ y^D \end{bmatrix}_{D\times 1} \qquad X = \begin{bmatrix} 1 & x^1 \\ \vdots & \vdots \\ 1 & x^D \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_N^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^D & x_2^D & \dots & x_N^D \end{bmatrix}_{D\times(1+N)},$$

then the the squared error is reduced by

$$J(w) = \|Xw - Y\|^2 .$$

Hence, the problem of finding an optimal weight vector for linear regression is a least squares problem.

Subsection 3

Denoising

Formulation

We observe a **noisy signal** $y \in \mathbb{R}^D$ of an unknown **true signal** $x \in \mathbb{R}^D$.
The goal is to reconstruct the smooth unknown x from a noisy signal y by balancing between

$$\text{acccuracy: } \|x - y\|^2 \qquad \text{and} \qquad \text{smoothness: } \sum_{t=1}^{D-1} (x_t - x_{t+1})^2 .$$

Finally, the optimization formulation of denoising is

$$\min_{x \in \mathbb{R}^D} \quad J(x) = \underbrace{\|x - y\|^2}_{J_1(x)} + \alpha \underbrace{\sum_{t=1}^{D-1} (x_t - x_{t+1})^2}_{J_2(x)},$$

where $\alpha > 0$ is the weight parameter.

Vectorization

The part $J_1(x)$ is already written in a vectorial manner (*i.e.* without expressing the components of each vectors).

On the other hand, the part $J_2(x)$ is defined explicitly in the components of x. To write it in a vectorized form, we introduce

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix}_{(D-1)\times D}$$

and note that

$$J_2(x) = \|Lx\|^2 .$$

We get

$$J(x) = \|x - y\|^2 + \alpha\|Lx\|^2 \quad \text{and} \quad \nabla J(x) = 2(x - y) + 2\alpha L^t Lx = 2(I + \alpha L^t L)x - 2y .$$

That's it!

Key takeaways.

- The implementation of different line search methods with gradient descent algorithm.
- The formulation and resolution of applications.

Thank you.