Gradient Descent

Martin Licht

UC San Diego

Winter Quarter 2021

Gradient Descent

We focus on systems $Ax = b$ with A symmetric positive definite.

We express this solution as an optimization problem.

Then we develop an ***iterative*** method for its solution, called ***gradient descent***.

Iterative in numerical linear algebra follow a different paradigm than classical methods such as Gaussian elimination.

Gradient Descent

Gradient Descent is an iterative numerical method for solving linear systems of equations.

More generally, there are gradient descent methods in optimization and machine learning, which are not part of this video.

Energy Functionals

Algorithm for Gradient Descent

Iterative Methods, general perspective

Energy Functionals

Let $A \in \mathbb{R}^{n \times n}$ be positive definite and $b \in \mathbb{R}^n$.

Let $x_\star \in \mathbb{R}^n$ be the unique solution to $Ax = b$.

For any $x \in \mathbb{R}^n$, thought of as an approximation of $x_\star$, we call $x_\star - x$ the ***error*** and $b - Ax$ the ***residual***.

Note that $b - Ax = A(x_\star - x)$.

Energy Functionals

We define the ***error functional***

$$E : \mathbb{R}^n \to \mathbb{R}$$

associated with A and b by setting

$$E(x) = \frac{1}{2}\langle x_\star - x, A(x_\star - x)\rangle.$$

Let's have a look at the properties of that functional.

Energy Functionals

Lemma

We have $E(x) = 0$ if and only if $x = x_\star$.

Proof.

The matrix A is symmetric positive definite. Hence $E(x) = 0$, that is,

$$\frac{1}{2}\langle x_\star - x, A(x_\star - x)\rangle = 0$$

if and only if $x_\star - x = 0$, that is, $x = x_\star$. □

Energy Functionals

Lemma

We have $E(x) > 0$ *whenever* $x \neq x_\star$.

Proof.

The matrix A is symmetric positive definite. Hence $E(x) > 0$, that is,

$$\frac{1}{2}\langle x_\star - x, A(x_\star - x)\rangle > 0$$

if and only if $x_\star - x \neq 0$, that is, $x \neq x_\star$. □

Energy Functionals

This shows that E has a global minimum at $x = x_\star$, and this minimum is only achieved at $x = x_\star$.

What about local minima? We will study where the gradient of E vanishes.

Energy Functionals

Lemma

The gradient of E at $x \in \mathbb{R}^n$ is $\nabla E(x) = b - Ax$.

Proof.

We compute

$$\begin{aligned} E(x) &= \frac{1}{2}\langle x_\star, Ax_\star\rangle - \frac{1}{2}\langle x, Ax_\star\rangle - \frac{1}{2}\langle x_\star, Ax\rangle + \frac{1}{2}\langle x, Ax\rangle \\ &= \frac{1}{2}\langle x_\star, Ax_\star\rangle - \frac{1}{2}\langle x, Ax_\star\rangle - \frac{1}{2}\langle Ax_\star, x\rangle + \frac{1}{2}\langle x, Ax\rangle \\ &= \frac{1}{2}\langle x_\star, Ax_\star\rangle - \langle x, b\rangle + \frac{1}{2}\langle x, Ax\rangle \end{aligned}$$

We calculate the gradient: $\nabla E(x) = b - Ax$. □

Energy Functionals

Lemma

The gradient of E vanishes only at $x = x_\star$.

Corollary

The only local minimum of E is the global minimum at $x_\star$.

Energy Functionals

The error functional $E(x)$ can be written as

$$E(x) = \frac{1}{2}x_\star, Ax_\star\rangle - \langle x, b\rangle + \frac{1}{2}\langle x, Ax\rangle$$

In applications, we know A and b but not $x_\star$. So we cannot compute $E(x)$.

However, the ***energy functional*** $F : \mathbb{R}^n \to \mathbb{R}$ that is given by

$$F(x) = \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle$$

differs from E only by a constant term $\frac{1}{2}\langle x_\star, Ax_\star\rangle$.

In particular, E and F have the same gradients and their minima are at the same place. F is computable in practice.

Energy Functionals

Thus, we already know that the energy functional

$$F(x) = \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle$$

has got only one single local minimum, namely a global minimum at $x = x_\star$.

We can use techniques of optimization to approximate the minimum of F.

In practice: make a first guess $x_{(0)}$ and compute an improved guess $x_{(1)}$. From that, compute a better guess $x_{(2)}$. From that, compute a better guess $x_{(3)}$. ....

Energy Functionals

Algorithm for Gradient Descent

Iterative Methods, general perspective

Algorithm for Gradient Descent

Suppose take a hike on the graph of F.

It's foggy and we don't see the whole landscape. But we see the immediate landscape around us.

Question: which direction should we go to move towards the bottom of the valley, towards the minimum of the energy functional?

Answer: the direction of steepest descent, that is, the negative gradient $-\nabla F(x)$.

Algorithm for Gradient Descent

If we are at x, we should walk into the direction $-\nabla F(x)$.

Of course, we should not walk indefinitely (we might just walk straight across the bottom of the valley and then up again...).

More specifically, we should walk into the direction of the negative gradient to lowest point possible.

Algorithm for Gradient Descent

Suppose that $x \in \mathbb{R}^n$ and let $d \in \mathbb{R}^n$ be ***any*** direction.

Consider the functional $f(t) = F(x + t \cdot d)$. Then

$$\begin{aligned} f(t) &= \frac{1}{2}\langle x + t \cdot d, A(x + t \cdot d)\rangle - \langle b, x + t \cdot d\rangle \\ &= \frac{1}{2}\langle x, Ax\rangle + t \cdot \langle d, Ax\rangle + \frac{t^2}{2}\langle d, Ad\rangle - \langle b, x\rangle - t \cdot \langle b, d\rangle \\ &= \frac{t^2}{2}\langle d, Ad\rangle + t \cdot \langle d, Ax - b\rangle + \frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle \end{aligned}$$

So this is a quadratic function in t.

Where is the minimum of that function?

Algorithm for Gradient Descent

We compute the derivative of that function:

$$f'(t) = t\langle d, Ad\rangle + \langle d, Ax - b\rangle.$$

This derivative is zero, $f'(t) = 0$, if and only if

$$\begin{aligned} 0 &= t\langle d, Ad\rangle + \langle d, Ax - b\rangle \\ \langle d, b - Ax\rangle &= t\langle d, Ad\rangle \\ \frac{\langle d, b - Ax\rangle}{\langle d, Ad\rangle} &= t. \end{aligned}$$

Hence that t tells us how much to walk from x in direction d to get to the minimum of the line through x in direction d.

Algorithm for Gradient Descent

Let us walk into the direction of the negative gradient,

$$d := -\nabla E(x) = b - Ax.$$

This is the residual at x.

The minimum of E over the line through x in direction d is found at $x + \alpha d$, where

$$\alpha = \frac{\langle d, b - Ax \rangle}{\langle d, Ad \rangle} = \frac{\langle d, d \rangle}{\langle d, Ad \rangle}, \quad d = b - Ax.$$

Let's repeat this over and over again ....

Algorithm for Gradient Descent

We start at $x_{(0)}$ and compute the next position $x_{(1)}$,

$$x_{(1)} = x_{(0)} + \alpha_{(0)} d_{(0)}, \quad \alpha_{(0)} = \frac{\langle d_{(0)}, d_{(0)} \rangle}{\langle d_{(0)}, A d_{(0)} \rangle}, \quad d_{(0)} = b - A x_{(0)}.$$

From here, compute the next position $x_{(2)}$, and from there, the next position $x_{(3)}$, and so on...

$$x_{(2)} = x_{(1)} + \alpha_{(1)} d_{(1)}, \quad \alpha_{(1)} = \frac{\langle d_{(1)}, d_{(1)} \rangle}{\langle d_{(1)}, A d_{(1)} \rangle}, \quad d_{(1)} = b - A x_{(1)},$$

$$x_{(3)} = x_{(2)} + \alpha_{(2)} d_{(2)}, \quad \alpha_{(2)} = \frac{\langle d_{(2)}, d_{(2)} \rangle}{\langle d_{(2)}, A d_{(2)} \rangle}, \quad d_{(2)} = b - A x_{(2)}.$$

Algorithm for Gradient Descent

The general formula is

$$x_{(k+1)} = x_{(k)} + \alpha_{(k)} d_{(k)}, \quad \alpha_{(k)} = \frac{\langle d_{(k)}, d_{(k)} \rangle}{\langle d_{(k)}, A d_{(k)} \rangle}, \quad d_{(k)} = b - A x_{(k)}.$$

The value of F becomes smaller and smaller, so we hope that the sequence $x_{(0)}, x_{(1)}, x_{(2)}, \ldots$ converges to the minimum at $x_\star$.

In practice, we stop the iteration after some stopping criterion. For example, maximum number of steps, size of residual, or difference between iterates ...

Algorithm for Descent

We outline the following pseudocode for gradient descent

1: **GradientDescent**
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad d = b - A \cdot x$
4: $\quad \alpha = \dfrac{\langle d, d \rangle}{\langle d, Ad \rangle}$
5: $\quad x = x + \alpha r$
6: **end for**

Algorithm for Descent

In order to save an matrix-vector multiplication per iteration, we introduce an auxiliary variable

1: **GradientDescent**
2: $d = b - A \cdot x$
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad p = A \cdot r$
5: $\quad \alpha = \dfrac{\langle d, d \rangle}{\langle d, p \rangle}$
6: $\quad x = x + \alpha r$
7: $\quad d = d - \alpha p$
8: **end for**

Gradient Descent

We can add stopping criteria at this point. For example, we impose a maximum iteration count and exit the loop when the residual norm is below a certain threshold.

1: **GradientDescent**
2: $d = b - A \cdot x$
3: **for** $k = 1, 2, \ldots, k_{\max}$ **do**
4: $\quad p = A \cdot r$
5: $\quad \alpha = \dfrac{\langle d, d \rangle}{\langle d, p \rangle}$
6: $\quad$ **if** $\langle d, d \rangle < \epsilon$ **then exit**
7: $\quad x = x + \alpha r$
8: $\quad d = d - \alpha p$
9: **end for**

Choosing stopping criteria includes some craft and guesswork, and is frequently treated with indifference.

Energy Functionals

Algorithm for Gradient Descent

Iterative Methods, general perspective

Iterative Methods, general perspective

We want to solve a linear system of equations $Ax = b$.

Direct methods, such as LU decomposition and Cholesky decomposition, compute the exact solution in a fixed number of steps.

Iterative methods, such as gradient descent, approximate the solution to the desired precision. There is no theoretical limit on the number of steps.

Iterative Methods, general perspective

Runtime

Direct methods typically have run-time of cubic order: the number of calculation steps is cubically in the system size n

Iterative methods have run-time quadratic in n (for matrix multiplication) and linear in the number of steps k. Might be much faster if only a few steps are taken, $k \ll n$.

Iterative Methods, general perspective

Runtime

Direct methods have the same run-time for all systems of the same size.

Iterative methods have run-times (until a certain precision is reached) that depend on matrix properties (e.g., eigenvalue distribution).

Iterative Methods, general perspective

Precision

Direct methods produce the exact result in exact arithmetic, in practice affected by rounding errors.

Iterative methods are inexact by nature, in addition to rounding errors. They achieve the desired level of accuracy by keeping on iterating.

In practice, the accuracy of these numerical methods depends on matrix properties and rounding errors.

Iterative Methods, general perspective

Memory

Direct methods do not respect sparsity structures of matrices. The intermediate memory consumption might be much more than the input date.

Iterative methods typically do not consume much more memory than the input.

Iterative Methods, general perspective

Choice of numerical methods depends on many practical considerations. Rule of Thumb:

- Direct methods are best for small systems, without any particular structure.
- Iterative methods are best for large systems with specific sparsity structure.