Modern Optimization Techniques

Lucas Rego Drumond

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Gradient Descent

Outline

1. Unconstrained Optimization

2. Line search

3. Gradient Descent

Outline

1. Unconstrained Optimization

2. Line search

3. Gradient Descent

Unconstrained Optimization Problems

An **unconstrained optimization problem** has the form:

$$\text{minimize} \quad f_0(\mathbf{x})$$

Where:

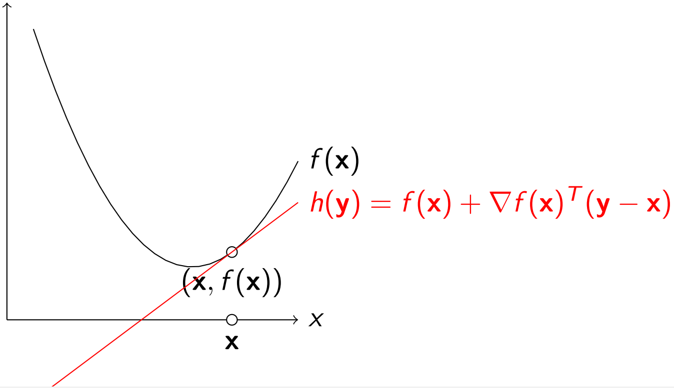
- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is convex, twice differentiable
- An optimal $\mathbf{x}^*$ exists and $f(\mathbf{x}^*)$ is attained and finite

1st-order condition

1st-order condition: a differentiable function f is convex iff

- dom f is a convex set
- for all $\mathbf{x}, \mathbf{y} \in \text{dom } f$

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



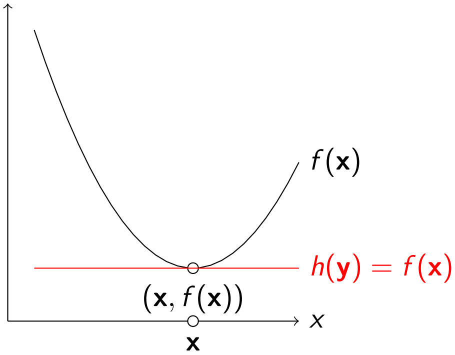
Optimality condition

f_0 is differentiable if dom f_0 is open and the gradient exists:

$$\nabla f_0(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right)$$

$\mathbf{x}^*$ is (localy) optimal iff:

$$\nabla f_0(\mathbf{x}) = 0$$



Methods for Unconstrained Optimization

- Start with an initial solution: $\mathbf{x}^0$
- Generate a sequence of points: $\mathbf{x}^t$ with

$$f_0(\mathbf{x}^t) \rightarrow f_0(\mathbf{x}^*)$$

procedure UNCONSTRAINEDMINIMIZATION
input: f_0
  Get initial point $\mathbf{x}^0$
  $t \leftarrow 1$
  repeat
    $\mathbf{x}^t \leftarrow \text{NextPoint}(\mathbf{x}^{t-1})$
    $t \leftarrow t + 1$
  until convergence
  return $\mathbf{x}^t$, $f_0(\mathbf{x}^t)$
end procedure

Convergence Criterion

- May depend on the optimization method
- Intuitively, one would use something like

$$\|\mathbf{x}^t - \mathbf{x}^\star\|_2^2 < \epsilon$$

- Since $\mathbf{x}^\star$ is unknown: $\|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2^2 < \epsilon$

Descent Methods

The next point is generated as:

$$\mathbf{x}^{t+1} := \mathbf{x}^t + \mu \Delta \mathbf{x}^t$$

Using:

- A step size μ
- A direction $\Delta \mathbf{x}$ such that

$$f_0(\mathbf{x}^t + \mu \Delta \mathbf{x}^t) < f_0(\mathbf{x}^t)$$

```
procedure DescentMethod
input: f_0
    Get initial point x
    t ← 0
    repeat
        Get Update Direction Δx
        Get Step Size μ
        x^{t+1} ← x^t + μΔx^t
        t ← t + 1
    until convergence
    return x, f_0(x)
end procedure
```

Descent Methods

- The descent algorithms differ in how they define the search direction $\Delta \mathbf{x}$
- The Step Size can be computed in various ways:
  - Fixed value
  - Line search
  - Various algorithm dependent heuristics

Outline

1. Unconstrained Optimization

2. Line search

3. Gradient Descent

Line search

Line Search is a practical method for computing the step lenght in descent algorithms

It solves the following problem for the variable μ:

$$\arg\min_{\mu>0} f_0(\mathbf{x} + \mu \Delta \mathbf{x})$$

Many variants of the line search:

- Exact throught derivation with respect to μ
- Approximative: e.g. Backtracking

Line Search

Exact

- Used if the problem can be solved analytically or with a low cost

Backtracking

- Only aproximative
- Guarantees that the new function value is lower than an specific bound

Backtracking Line Search

1: **procedure** BACKTRACKINGLINESEARCH
   input: f_0, search direction $\Delta\mathbf{x}$, at $\mathbf{x}$, $a \in (0, 0.5)$, $b \in (0, 1)$
2: $\mu \leftarrow 1$
3: **while** $f_0(\mathbf{x} + \mu\Delta\mathbf{x}) > f_0(\mathbf{x}) + a\mu\nabla f_0(\mathbf{x})^T\Delta\mathbf{x}$ **do**
4: $\mu \leftarrow b\mu$
5: **end while**
6: **return** μ
7: **end procedure**

Outline

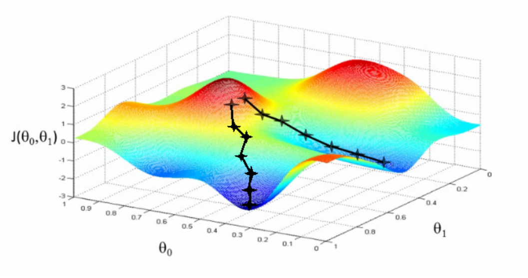
Gradient Descent

- The gradient of a function $f : \mathbb{R} \to \mathbb{R}^n$ in $\mathbf{x}$ shows the direction in which the function is maximally growing at point $\mathbf{x}$
- Gradient Descent is a descent algorithm that searches in the opposite direction of the gradient

$$\Delta\mathbf{x} = -\nabla f_0(\mathbf{x})$$

Gradient Descent

procedure GRADIENTDESCENT
input: f_0
    Get initial point $\mathbf{x}$
    repeat
        Get Step Size μ
        $\mathbf{x} := \mathbf{x} - \mu \nabla f_0(\mathbf{x})$
    until convergence
    return $\mathbf{x}$, $f_0(\mathbf{x})$
end procedure



Gradient Descent - Considerations

- Stopping criterion: $||\nabla f_0(\mathbf{x})||_2 \leq \epsilon$
- Simple and straightforward
- Usually slow convergence
- Works only well for convex problems, otherwise gets stuck in local minima
- Rarely used on practice

Gradient Descent Example

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f_0(\mathbf{x}) = -2\mathbf{x}$

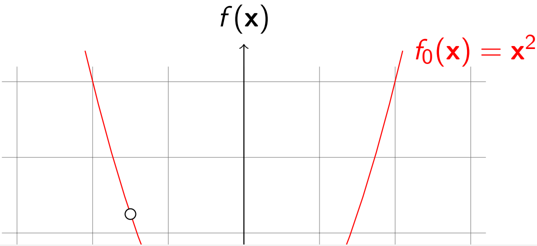
Initial point: $\mathbf{x}^0 = -1.5$

$$\mathbf{x}^0 = -1.5$$

$$\mathbf{x} = -1.5 - 0.3 \cdot (2 \cdot -1.5)$$

$$\mathbf{x} = -0.6$$

$$\mathbf{x} = -0.6$$



Considerations about the Step Size

- Crucial for the convergence of the algorithm
- Step size too low $\implies$ slow convergence
- Step size too high $\implies$ divergence!

Gradient Descent Example - A perfect Step Size

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

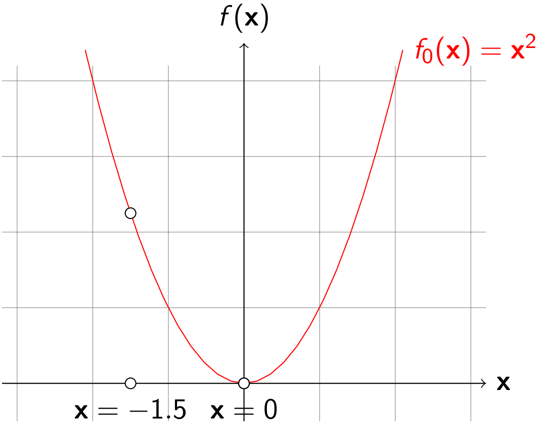
- $\mu = 0.5$
- $-\nabla f_0(\mathbf{x}) = -2\mathbf{x}$

Initial point: $\mathbf{x}^0 = -1.5$

$$\mathbf{x}^0 = -1.5$$

$$\mathbf{x} = -1.5 - 0.5 \cdot (2 \cdot -1.5)$$

$$\mathbf{x} = 0$$



Gradient Descent Example - Too High Step Size

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 1.5$
- $-\nabla f_0(\mathbf{x}) = -2\mathbf{x}$

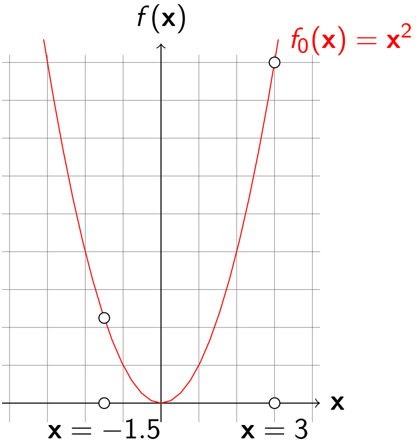
Initial point: $\mathbf{x}^0 = -1.5$

$\mathbf{x}^0 = -1.5$

$\mathbf{x} = -1.5 - 1.5 \cdot (2 \cdot -1.5)$

$\mathbf{x} = 3$

$\mathbf{x}^0 = 3$



A More practical example

We do not want to always minimize parabolas so let us discuss a more practical example:

Linear Regression!

- have m many data instances $\mathbf{a} \in \mathbb{R}^n$ with n many features / predictors
- want to learn a linear model parametrized by a vector $\beta \in \mathbb{R}^n$ to predict a real value $y \in \mathbb{R}$

Practical Example: Household Spending

If we have data about m households, we can represent it as:

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can model the household consumption is a linear combination of the household features with parameters β:

$$\hat{y}_i = \beta^T \mathbf{a_i} = \beta_0 1 + \beta_1 a_{i,1} + \beta_2 a_{i,2} + \beta_3 a_{i,3} + \beta_4 a_{i,4}$$

Practical Example: Household Spending

We have:

$$\begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \approx \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We want to find parameters β such that the measured error of the predictions is minimal:

$$\sum_{i=1}^{m} (\beta^T \mathbf{a_i} - y_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2 = \|A\beta - y\|_2^2 + \lambda \|\beta\|_2^2$$

Linear Regression

Let us look at the function to optimize:

$$\begin{aligned}\mathcal{L}(\beta, A, y) + \lambda \text{Reg}(\beta) &= \sum_{i=1}^{m} (\beta^\top a_i - y_i)^2 + \lambda \|\beta\|_2^2 \\ &= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right)^2 + \lambda \sum_{j=1}^{n} \beta_j^2\end{aligned}$$

Then we can compute the gradient component wise:

$$\begin{aligned}\frac{\partial}{\partial \beta_k} \mathcal{L}(\beta, A, y) + \lambda \text{Reg}(\beta) &= \frac{\partial}{\partial \beta_k} \sum_{i=1}^{m} (\sum_{j=1}^{n} \beta_j a_{ij} - y_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2 \\ &= \sum_{i=1}^{m} 2 \cdot \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right) \cdot a_{ik} + 2\lambda\beta_k\end{aligned}$$

Linear Regression

We obtain the update for every component of β as

$$\begin{aligned}\beta_k^{t+1} &= \beta_k^t - \mu \nabla_\beta (\mathcal{L}(\beta, A, y) + \lambda \mathrm{Reg}(\beta)) \\ &= \beta_k^t - \mu \left(2 \sum_{i=1}^{m} \cdot \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right) \cdot a_{ik} + 2\lambda \beta_k^t \right)\end{aligned}$$

- see that $\left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i\right)$ is actually the error of the model on the i-th instance
- error is the same for all k, can be precomputed

Linear Regression

procedure LEARN LINEAR REGRESSION MODEL
input: Data A, Labels y, inital parameters β^0, Step Size μ, Regularization constant λ, precision ϵ
  repeat
    Compute Error: $e_i = \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right)$
    for $k = 1, \ldots, n$ **do**
      $\beta_k^{t+1} = \beta_k^t - \mu \left(\sum_{i=1}^{m} e_i a_{ik} + \lambda \beta_k^t \right)$
    end for
    $t = t + 1$
  until $\|\nabla_\beta \mathcal{L}(\beta, A, y)\|_2^2 \leq \epsilon$
  return β, $\mathcal{L}(\beta, A, y)$
end procedure

Outlook

We will see in the next lectures:

- Stochastic Gradient Descent
  - Gradient is only computed on one instance, not on all
- Coordinate Descent
  - β is optimized in each coordinate
- Newton's Method
  - involves second order derivatives (curvature) information