COMPSCI 689
Lecture 5: Numerical Optimization Foundations

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Empirical Risk Minimization

Let Θ be the parameter space for a set $\mathcal{F}$ of prediction functions f_θ mapping from $\mathcal{X}$ to $\mathcal{Y}$. The principle of Empirical Risk Minimization (ERM) states that we should select the parameters $\theta \in \Theta$ that *minimize the average of the prediction loss* $L(\mathbf{y}_n, f_\theta(\mathbf{x}_n))$ computed over the data set $\mathcal{D}$, also known as the empirical risk $R(f_\theta, \mathcal{D})$:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} R(f_\theta, \mathcal{D})$$

$$R(f_\theta, \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} L(\mathbf{y}_n, f_\theta(\mathbf{x}_n))$$

ERM and Optimization

In a general ERM-based learning problem:

- The optimization problem is a minimization problem
- The model parameters are the optimization variables
- The risk function is the objective function
- The objective function may or may not be differentiable
- The optimization problem may or may not have constraints

Notation

- In this lecture we will follow a standard presentation for numerical optimization.
- We will assume $\mathbf{x}$ are the variables to be optimized.
- We will assume $f(\mathbf{x})$ is the function to be optimized.
- *Note that the definitions of* $\mathbf{x}$ *and* f *differ from those used in the previous lectures on supervised learning.*

An Optimization Problem

An optimization problem in standard form consists of four primary components:

$$\min_{x \in \mathcal{X}} f(\mathbf{x}) \qquad \text{subject to} \begin{cases} c_i(\mathbf{x}) = 0, & i \in \mathcal{E} \\ c_i(\mathbf{x}) \geq 0, & i \in \mathcal{I} \end{cases}$$

- Variables: $\mathbf{x} = [x_1, ..., x_M] \in \mathcal{X}$
- Objective Function: $f : \mathcal{X} \rightarrow \mathbb{R}$
- Equality Constraints: $c_i(\mathbf{x}) = 0$, $i \in \mathcal{E}$
- Inequality Constraints: $c_i(\mathbf{x}) \geq 0$, $i \in \mathcal{I}$

Types of Optimization Problems

There are several different categories of optimization problems distinguished by the following characteristics:

- Constrained vs Unconstrained
- Discrete vs Continuous vs Mixed Variables
- Differentiable vs Non-Differentiable Objective Function
- Convex vs Non-Convex Objective Function

Convex Optimization

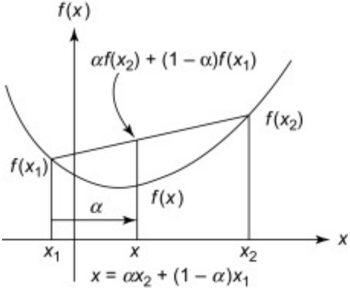
A function $f : \mathcal{X} \rightarrow \mathcal{R}$ is said to be convex if $\mathcal{X}$ is a convex set and for any two points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and any scalar $\alpha \in [0, 1]$:

$$f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{x}') \leq \alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{x}')$$

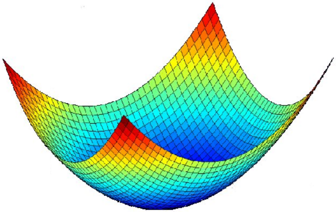
A set $\mathcal{X}$ is convex if for any two points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and any scalar $\alpha \in [0, 1]$:

$$\alpha\mathbf{x} + (1 - \alpha)\mathbf{x}' \in \mathcal{X}$$

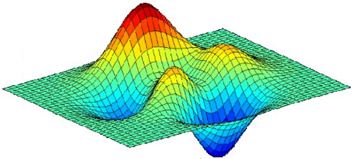
Convex Function


$f(x)$
$\alpha f(x_2) + (1-\alpha)f(x_1)$
$f(x_2)$
$f(x_1)$
$f(x)$
α
x
x_1
x
x_2
$x = \alpha x_2 + (1-\alpha)x_1$

Convex Function

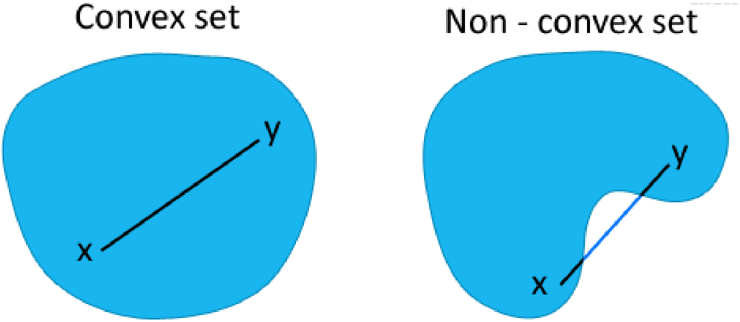


convex function



non-convex function

Convex Function



Convex Functions and Sets

Useful examples of convex functions:

- Constant functions
- Linear functions
- Quadratic functions that are positive semi-definite
- The squared loss, absolute loss, and logistic loss
- A sum of convex functions
- A weighted sum of convex functions with non-negative weights

Useful examples of convex sets:

- $\mathbb{R}^M$
- The probability simplex

Types of Solutions

Key definitions of types of optimization solutions:

- **Global Minimizer:** A point $\mathbf{x}_* \in \mathcal{X}$ is a global minimizer of f if and only if $f(\mathbf{x}_*) \leq f(\mathbf{x})$ for all $x \in \mathcal{X}$.
- **Local Minimizer:** A point $\mathbf{x}_* \in \mathcal{X}$ is a local minimizer of f if and only if $f(\mathbf{x}_*) \leq f(\mathbf{x})$ for all x in an open set around $\mathbf{x}_*$.
- For most complex, non-convex functions, we can only hope to identify a local minima with no guarantee about it's quality relative to the global minimum.

Characterizing Solutions: Definitions

Key definitions for characterizing optimization solutions:

- **Gradient**: The gradient $\nabla f(\mathbf{x})$ of function $f(\mathbf{x})$ is the vector of partial derivatives of f with respect to each of the optimization variables: $[\nabla f(\mathbf{x})]_i = \frac{\partial}{\partial x_i} f(\mathbf{x})$.
- **Hessian**: The hessian $\nabla^2 f(\mathbf{x})$ of function $f(\mathbf{x})$ is the matrix of mixed partial derivatives of f with respect to each pair of optimization variables: $[\nabla^2 f(\mathbf{x})]_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x})$.
- **Stationary Point**: $\mathbf{x}$ is said to be a stationary point of f if $\nabla f(\mathbf{x}) = 0$.
- **Local Minimizer:** $\mathbf{x}$ is a local minimizer of f if $\mathbf{x}$ is a stationary point of f and the Hessian of f at $\mathbf{x}$ is positive semi-definite.
- **Convexity Condition**: If f is convex and differentiable and $\mathbf{x}_*$ is a stationary point of f, then $\mathbf{x}_*$ is a global minimizer of f

Finding Solutions

- Some optimization problems can be solved analytically by solving the system of gradient equations $\nabla f(\mathbf{x}) = 0$ to identify one or more stationary points $\mathbf{x}_*$, and then checking that $\nabla^2 f(\mathbf{x}_*)$ is positive definite (or appealing to convexity).
- When an analytic approach fails, an iterative numerical approach can be used to approximately locate a stationary point.

Iterative Optimization

- The key idea in iterative numerical optimization is to produce a sequence of iterates $\mathbf{x}_0, \mathbf{x}_1, \ldots$ starting from an initial guess $\mathbf{x}_0$.
- On each iteration k, we identify a direction $\mathbf{d}_k$ in which f decreases and then take a "step" of length $\alpha > 0$ in that direction to form the new iterate $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{d}_k$.

Finding Solutions

- A key result is that $-\nabla f(\mathbf{x}_k)$ is the direction of steepest decrease in f at $\mathbf{x}_k$.

- Further, for any direction $\mathbf{d}_k$ that has a positive inner product with $-\nabla f(\mathbf{x}_k)$, there exists an α that will decrease f.

- The key problems are how to choose $\mathbf{d}_k$ and how to choose α.

Steepest Descent with Fixed Step Sizes

This algorithm assumes we take $\mathbf{d}_k$ to be the steepest descent direction and we fix α to a constant value.

1. Set $\mathbf{x}_0$, $\alpha > 0$,
2. On each iteration k:
    1. Compute $\mathbf{d}_k$
    2. $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{d}_k$

The problem with this simple method is that it may not converge!

Backtracking Line Search

Backtracking Line Search is a simple method to choose α while monotonically decreasing f.

```
1 Set $\mathbf{x}_0$, $\alpha_0 > 0$, $0 < \rho < 1$
2 On each iteration $k$:
    1 $f_k \leftarrow f(\mathbf{x}_k)$
    2 Compute $\mathbf{d}_k$
    3 $\alpha \leftarrow \alpha_0$
    4 While $f(\mathbf{x}_k + \alpha\mathbf{d}_k) \geq f_k : \alpha \leftarrow \rho\alpha$
    5 $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha\mathbf{d}_k$
```

To ensure that the algorithm makes sufficient progress on each step, the Armijo rule is used: $f(\mathbf{x}_k + \alpha\mathbf{d}_k) \leq f_k + c\alpha d_k^T \nabla f(\mathbf{x}_k)$ for $0 < c \leq 1$.

Common choices are $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$, $\alpha_0 = 1$, $\rho = 0.5$.

The Newton Direction

- The Newton direction $\mathbf{d}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$ is the optimal descent direction for a quadratic function. Such a method is often called a “Newton” or “second-order” method since it uses the Hessian and not just the gradient (gradient methods are referred to as “first-order” methods).

- Using the Newton direction to optimize convex functions is typically more efficient than using first-order methods in terms of the number of iterations required.

- However, the overall run-time time for a Newton method can be higher due to the need to compute and invert the hessian matrix on every iteration. Newton methods also need quadratic storage, which can be prohibitive for large-scale problems.

Quasi-Newton Methods

- Quasi-Newton methods attempt to preserve the convergence benefits of Newton methods while reducing the run time and storage requirements.

- The general form of the descent direction is $\mathbf{d}_k = -(\mathbf{B}_k)^{-1}\nabla f(\mathbf{x})$ for a positive definite matrix $\mathbf{B}_k$.

- One of the more robust methods of this type is the Limited Memory BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, which iteratively constructs a low-rank, positive definite approximation to the hessian matrix. It can be used regardless of whether a function is (locally) convex.

- For many mid-scale problems, L-BFGS will out-perform the use of the steepest descent direction. It's generally a good choice when computationally feasible.

Convergence Assessment

- Convergence of numerical optimization methods is typically assessed either by looking at convergence of the gradient, the function values, or the iterates.
- A gradient condition will look like $\|\nabla f(\mathbf{x})\|_p \leq \tau_1$ for some p norm.
- A function value condition will look like $\frac{|f(\mathbf{x}_k)-f(\mathbf{x}_{k-1})|}{|f(\mathbf{x}_{k-1})|} \leq \tau_2$.
- An iterate condition will look like $\frac{\|\mathbf{x}_k-\mathbf{x}_{k-1}\|_p}{\|\mathbf{x}_{k-1}\|_p} \leq \tau_3$.
- A given method may use a mix of conditions. Incorrectly setting convergence parameters can result in the optimizer stopping too early, or running longer than needed.