最优化：简介

文再文

北京大学北京国际数学研究中心

教材《最优化：建模、算法与理论》配套电子教案

http://bicmr.pku.edu.cn/~wenzw/optbook.html

致谢：本教案由邓展望协助准备

提纲

最优化问题的一般形式

最优化问题一般可以描述为

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{1}$$

- $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$是决策变量
- $f: \mathbb{R}^n \to \mathbb{R}$ 是目标函数
- $\mathcal{X} \subseteq \mathbb{R}^n$是约束集合或可行域，可行域包含的点称为可行解或可行点. 记号 s.t. 是“subject to”的缩写，专指约束条件.
- 当$\mathcal{X} = \mathbb{R}^n$时，问题(1) 称为无约束优化问题.
- 集合$\mathcal{X}$通常可以由约束函数$c_i(x): \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \cdots, m + l$ 表达为如下具体形式：

$$\begin{aligned} \mathcal{X} = \{x \in \mathbb{R}^n \mid & c_i(x) \le 0, \quad i = 1, 2, \cdots, m, \\ & c_i(x) = 0, \quad i = m+1, m+2 \cdots, m+l\}. \end{aligned}$$

最优化问题的一般形式

- 在所有满足约束条件的决策变量中，使目标函数取最小值的变量x^* 称为优化问题(1) 的最优解，即对任意$x \in \mathcal{X}$ 都有

$$f(x) \geq f(x^*).$$

- 如果求解在约束集合$\mathcal{X}$ 上目标函数$f(x)$的最大值，则问题(1)的“min”应相应地替换为“max”.

- 注意到在集合$\mathcal{X}$ 上，函数f 的最小（最大）值不一定存在，但是其下（上）确界“inf f(sup f)” 总是存在的. 因此，当目标函数的最小(最大）值不存在时，我们便关心其下（上）确界，即将问题(1) 中的“min(max)” 改为“inf(sup)”.

- 为了叙述简便，问题(1) 中x为$\mathbb{R}^n$空间中的向量. 实际上，根据具体应用和需求，x还可以是矩阵、多维数组或张量等.

最优化问题的类型

最优化问题的具体形式非常丰富，可以按照目标函数、约束函数以及解的性质将其分类.

- 当目标函数和约束函数均为线性函数时，问题称为线性规划；
- 当目标函数和约束函数中至少有一个为非线性函数时，相应的问题称为非线性规划；
- 如果目标函数是二次函数而约束函数是线性函数则称为二次规划；
- 包含非光滑函数的问题称为非光滑优化；
- 不能直接求导数的问题称为无导数优化；
- 变量只能取整数的问题称为整数规划；
- 在线性约束下极小化关于半正定矩阵的线性函数的问题称为半定规划，其广义形式为锥规划.

最优化问题的类型

- 最优解只有少量非零元素的问题称为稀疏优化；
- 最优解是低秩矩阵的问题称为低秩矩阵优化.
- 此外还有几何优化、二次锥规划、张量优化、鲁棒优化、全局优化、组合优化、网络规划、随机优化、动态规划、带微分方程约束优化、微分流形约束优化、分布式优化等.
- 就具体应用而言，问题(1) 可涵盖统计学习、压缩感知、最优运输、信号处理、图像处理、机器学习、强化学习、模式识别、金融工程、电力系统等领域的优化模型.

最优化问题的应用

数学建模很容易给出应用问题不同的模型，可以对应性质很不相同的问题，其求解难度和需要的算法也将差别很大. 在投资组合优化中，人们希望通过寻求最优的投资组合以降低风险、提高收益.

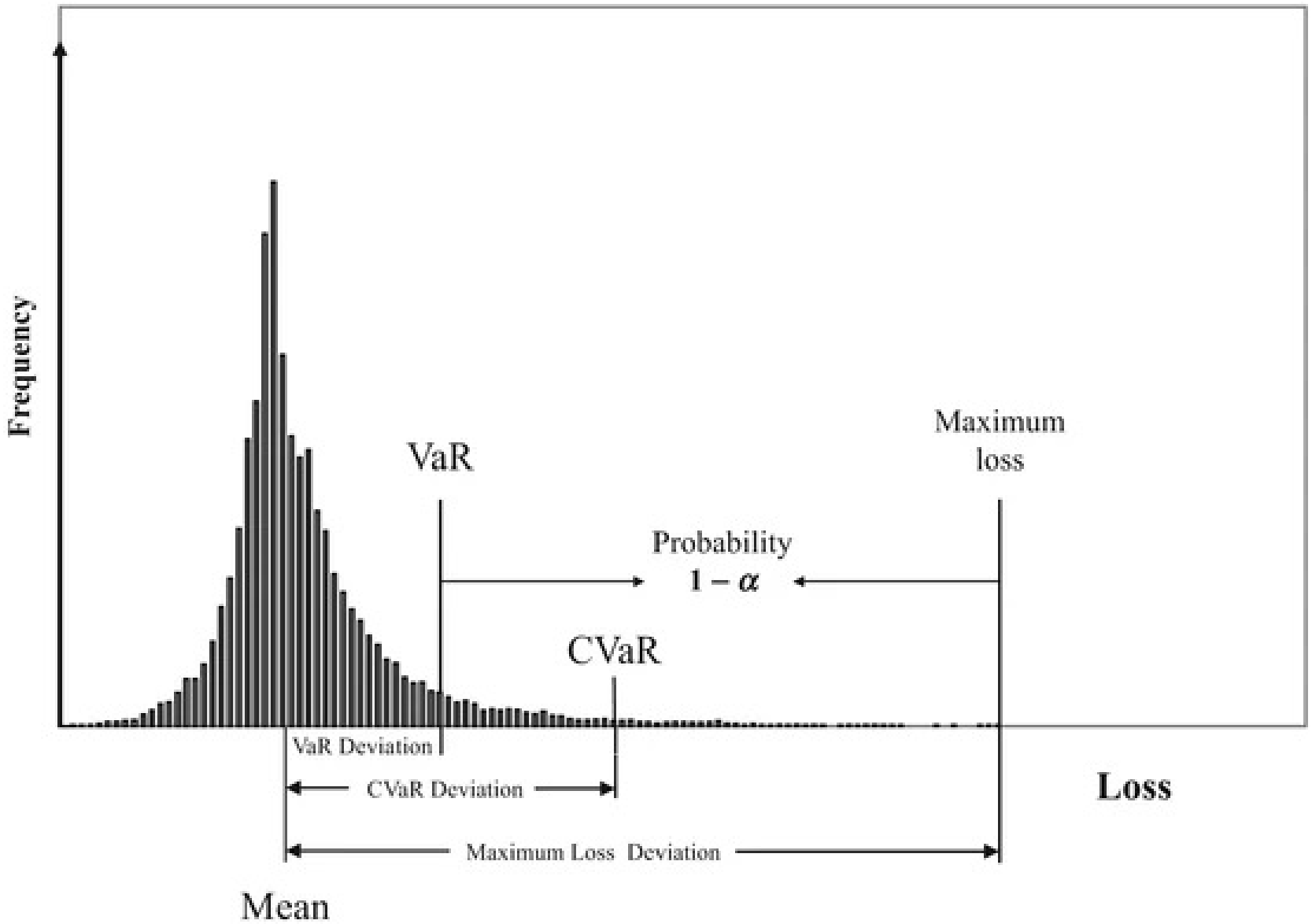
- 这时决策变量x_i表示在第i项资产上的投资额，向量$x \in \mathbb{R}^n$表示整体的投资分配.
- 约束条件可能为总资金数、每项资产的最大（最小）投资额、最低收益等.
- 目标函数通常是某种风险度量.
- 如果是极小化收益的方差，则该问题是典型的二次规划.
- 如果极小化风险价值(value at risk)函数，则该问题是混合整数规划
- 如果极小化条件风险价值（conditional value at risk)函数，则该问题是非光滑优化，也可以进一步化成线性规划.

投资组合优化

- r_i, 随机变量, 股票的回报率i
- x_i, 投资于股票的相对金额i
- 回报: $r = r_1x_1 + r_2x_2 + \ldots + r_nx_n$
- 期望回报: $R = E(r) = \sum E(r_i)x_i = \sum \mu_i x_i$
- 风险: $V = Var(r) = \sum_{i,j} \sigma_{ij}x_ix_j = x^\top \Sigma x$

$$\begin{aligned} \min \ & \frac{1}{2}x^\top \Sigma x, \\ \text{s.t.} \ & \sum \mu_i x_i \geq r_0 \\ & \sum x_i = 1, \\ & x_i \geq 0 \end{aligned} \qquad \begin{aligned} \min \ & \text{risk measure}, \\ \text{s.t.} \ & \sum \mu_i x_i \geq r_0 \\ & \sum x_i = 1, \\ & x_i \geq 0 \end{aligned}$$

风险度量


Frequency
VaR
Maximum
loss
Probability
1 – α
CVaR
VaR Deviation
CVaR Deviation
Loss
Maximum Loss Deviation
Mean

提纲

$$(\ell_0)\begin{cases}\min\limits_{x\in\mathbb{R}^n} & \|x\|_0,\\ \text{s.t.} & Ax=b.\end{cases}\quad(\ell_2)\begin{cases}\min\limits_{x\in\mathbb{R}^n} & \|x\|_2,\\ \text{s.t.} & Ax=b.\end{cases}\quad(\ell_1)\begin{cases}\min\limits_{x\in\mathbb{R}^n} & \|x\|_1,\\ \text{s.t.} & Ax=b.\end{cases}\tag{2}$$

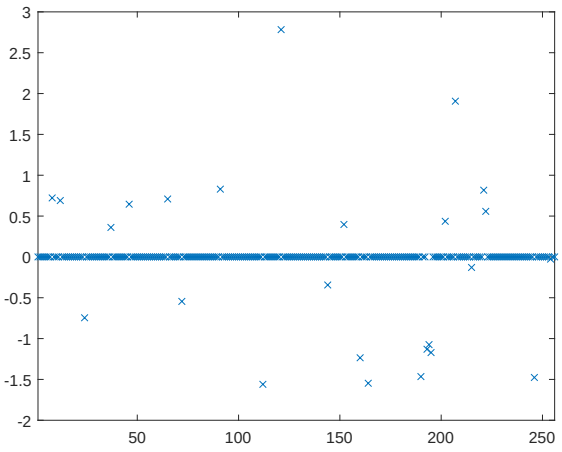
- 其中$\|x\|_0$是指x 中非零元素的个数. 由于$\|x\|_0$是不连续的函数，且取值只可能是整数，ℓ_0问题实际上是NP难的，求解起来非常困难.
- 若定义ℓ_1范数：$\|x\|_1=\sum_{i=1}^n|x_i|$，则得到了另一个形式上非常相似的问题，又称$\ell_1$ 范数优化问题，基追踪问题
- ℓ_2范数：$\|x\|_2=\left(\sum_{i=1}^n x_i^2\right)^{1/2}$

稀疏优化

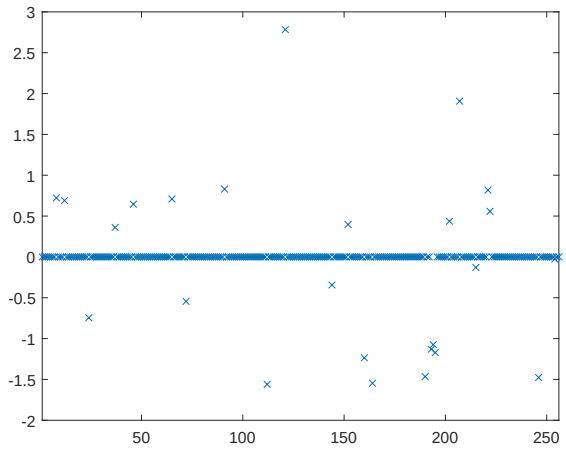
在MATLAB环境里构造A, u 和b：

```
m = 128; n = 256;
A = randn(m, n);
u = sprandn(n, 1, 0.1);
b = A * u;
```

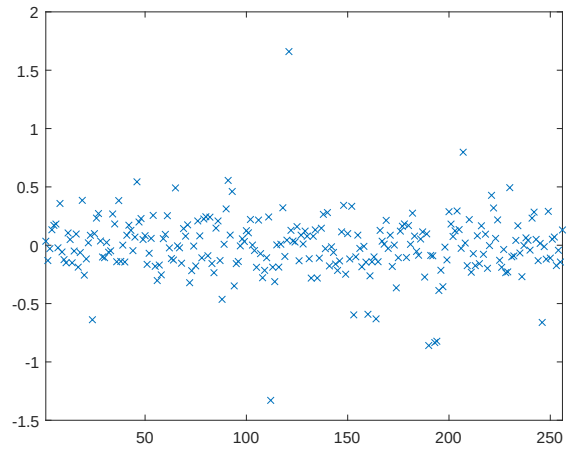
构造一个128×256矩阵A，它的每个元素都服从高斯（Gauss）随机分布．精确解u只有10%的元素非零，每一个非零元素也服从高斯分布



(a) 精确解u



(b) ℓ_1问题的解



(c) ℓ_2问题的解

Figure: 稀疏优化的例子

LASSO问题

考虑带ℓ_1范数正则项的优化问题

$$\min_{x\in\mathbb{R}^n} \quad \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2, \tag{3}$$

其中$\mu > 0$是给定的正则化参数.

- 问题(3) 又称为LASSO（least absolute shrinkage and selection operator），该问题可以看成是问题(2) 的二次罚函数形式.
- 由于它是无约束优化问题，形式上看起来比问题(2) 简单. 课件关注的大部分数值算法都将针对问题(2) 或问题(3) 给出具体形式. 因此全面掌握它们的求解方法是掌握基本最优化算法的一个标志.

提纲

低秩矩阵恢复

- 某视频网站提供了约48万用户对1万7千多部电影的上亿条评级数据，希望对用户的电影评级进行预测，从而改进用户电影推荐系统，为每个用户更有针对性地推荐影片.

- 显然每一个用户不可能看过所有的电影，每一部电影也不可能收集到全部用户的评级. 电影评级由用户打分1 星到5 星表示，记为取值1~5 的整数. 我们将电影评级放在一个矩阵M中，矩阵M的每一行表示不同用户，每一列表示不同电影. 由于用户只对看过的电影给出自己的评价，矩阵M中很多元素是未知的

	电影1	电影2	电影3	电影4	…	电影n
用户1	4	?	?	3	…	?
用户2	?	2	4	?	…	?
用户3	3	?	?	?	…	?
用户4	2	?	5	?	…	?
⋮	⋮	⋮	⋮	⋮		⋮
用户m	?	3	?	4	…	?

低秩矩阵恢复问题的性质

该问题在推荐系统、图像处理等方面有着广泛的应用。

- 由于用户对电影的偏好可进行分类，按年龄可分为：年轻人，中年人，老年人；且电影也能分为不同的题材：战争片，悬疑片，言情片等。故这类问题隐含的假设为补全后的矩阵应为低秩的。即矩阵的行与列会有“合作”的特性，故该问题具有别名“collaborative filtering”。
- 除此之外，由于低秩矩阵可分解为两个低秩矩阵的乘积，所以低秩限制下的矩阵补全问题是比较实用的，这样利于储存且有更好的诠释性。
- 有些用户的打分可能不为自身真实情况，对评分矩阵有影响，所以原矩阵是可能有噪声的。

低秩矩阵恢复

由上述分析可以引出该问题：

- 令Ω是矩阵M中所有已知评级元素的下标的集合，则该问题可以初步描述为构造一个矩阵X，使得在给定位置的元素等于已知评级元素，即满足$X_{ij} = M_{ij},\ (i,j) \in \Omega$.

- 低秩矩阵恢复（low rank matrix completion）

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & \operatorname{rank}(X), \\ \text{s.t.} \quad & X_{ij} = M_{ij},\ (i,j) \in \Omega. \end{aligned} \tag{4}$$

  $\operatorname{rank}(X)$正好是矩阵X所有非零奇异值的个数

- 矩阵X的核范数（nuclear norm）为矩阵所有奇异值的和，即：$\|X\|_* = \sum_i \sigma_i(X)$:

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & \|X\|_*, \\ \text{s.t.} \quad & X_{ij} = M_{ij},\ (i,j) \in \Omega. \end{aligned} \tag{5}$$

低秩矩阵恢复

- 可以证明问题(5) 是一个凸优化问题，并且在一定条件下它与问题(4) 等价.
- 也可以将问题(5) 转换为一个半定规划问题，但是目前半定规划算法所能有效求解的问题规模限制了这种技术的实际应用.
- 考虑到观测可能出现误差，对于给定的参数$\mu > 0$，给出该问题的二次罚函数形式：

$$\min_{X \in \mathbb{R}^{m \times n}} \quad \mu \|X\|_* + \frac{1}{2} \sum_{(i,j) \in \Omega} (X_{ij} - M_{ij})^2. \tag{6}$$

低秩矩阵恢复

- 秩r情形：$X = LR^T$,其中$L \in \mathbb{R}^{m\times r}$,$R \in \mathbb{R}^{n\times r}$ 并且$r \ll \min(m,n)$.则可将问题写为

$$\min_{\boldsymbol{L},\boldsymbol{R}} \sum_{(i,j)\in\omega} \left(\left[\boldsymbol{L}\boldsymbol{R}^T\right]_{ij} - \boldsymbol{M}_{ij} \right)^2 + \alpha \|\boldsymbol{L}\|_F^2 + \beta \|\boldsymbol{R}\|_F^2$$

- 在该问题中，矩阵X在定义中已为秩r矩阵，所以没有必要再加上秩约束正则项。α, β为正则化参数，这里正则化的作用是消除解L, R在放缩意义下的不唯一性.

- 此时L, R矩阵中的数字之和为$(m+n)r$,远小于np,不过此时问题是非凸的.

- 尽管这个该问题是非凸的，但在某种意义上它是一个可处理问题的近似：如果对X 有一个完整的观察，那么秩-r近似可以通过X的奇异值分解来找到，并根据r导出的左奇异向量和右奇异向量定义L和R。

提纲

机器学习中典型问题形式

很多机器学习中的问题可以写为：

$$\min_{x\in\mathcal{W}} \quad \sum_{i=1}^{N} \frac{1}{2}\|a_i^\top x - b_i\|_2^2 + \mu\varphi(x) \quad \text{线性回归}$$

$$\min_{x\in\mathcal{W}} \quad \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-b_i a_i^\top x)) + \mu\varphi(x) \quad \text{逻辑回归}$$

$$\min_{x\in\mathcal{W}} \quad \frac{1}{N}\sum_{i=1}^{N} \ell(f(a_i,x),b_i) + \mu\varphi(x) \quad \text{一般形式}$$

- (a_i, b_i) 是给定的数据对，b_i 是数据a_i对应的标签
- $\ell_i(\cdot)$: 度量模型拟合数据点i的程度(避免拟合不足)
- $\varphi(x)$: 避免过拟合的正则项: $\|x\|_2^2$ 或者$\|x\|_1$等等
- $f(a,x)$: 线性函数或者由深度神经网络构造的模型

上述过程可用下图表示:

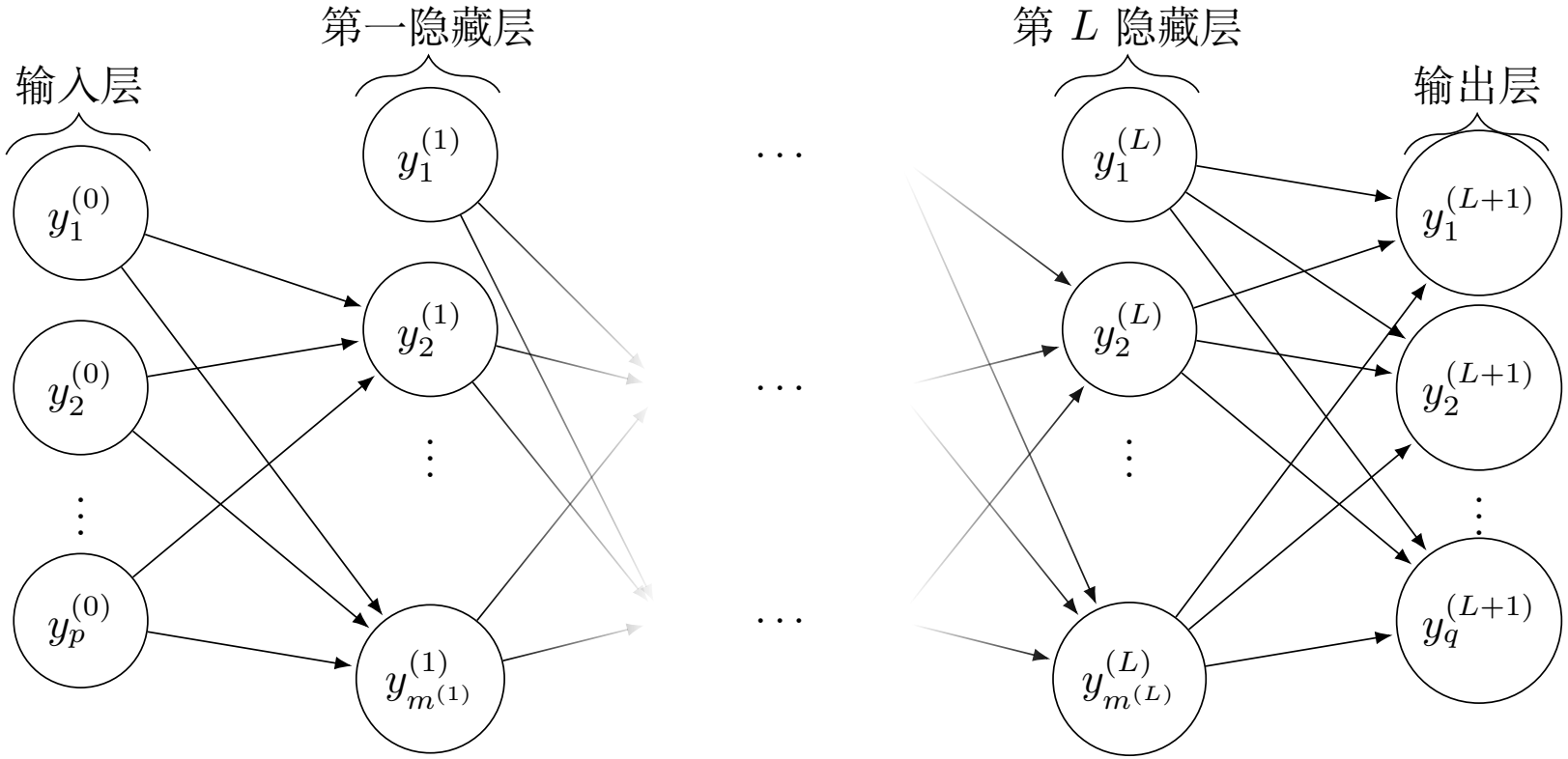


Figure: 带p个输入单元和q个输出单位的$(L+2)$层感知机的网络图，第l个隐藏层包含$m^{(l)}$个神经元.

卷积神经网络Convolutional neural network (CNN)

- 给定二维图像$\boldsymbol{I} \in \mathbb{R}^{n\times n}$和卷积核$\boldsymbol{K} \in \mathbb{R}^{k\times k}$，定义卷积操作$\boldsymbol{S} = \boldsymbol{I} * \boldsymbol{K}$，它的元素是

$$\boldsymbol{S}_{i,j} = \langle \boldsymbol{I}(i : i+k-1, j : j+k-1), \boldsymbol{K} \rangle,$$

其中两个矩阵X, Y的内积是它们相应元素乘积之和

- 生成的结果$\boldsymbol{S}$可以根据卷积核的维数、$\boldsymbol{I}$的边界是否填充、卷积操作时滑动的大小等相应变化.

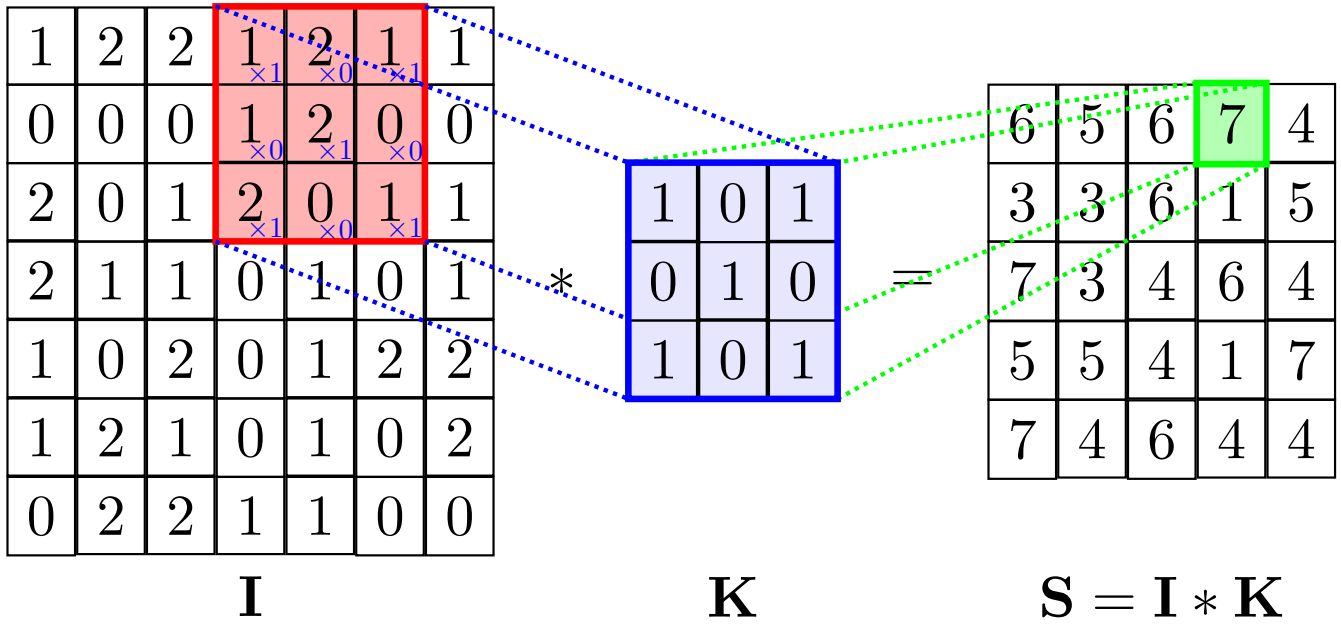
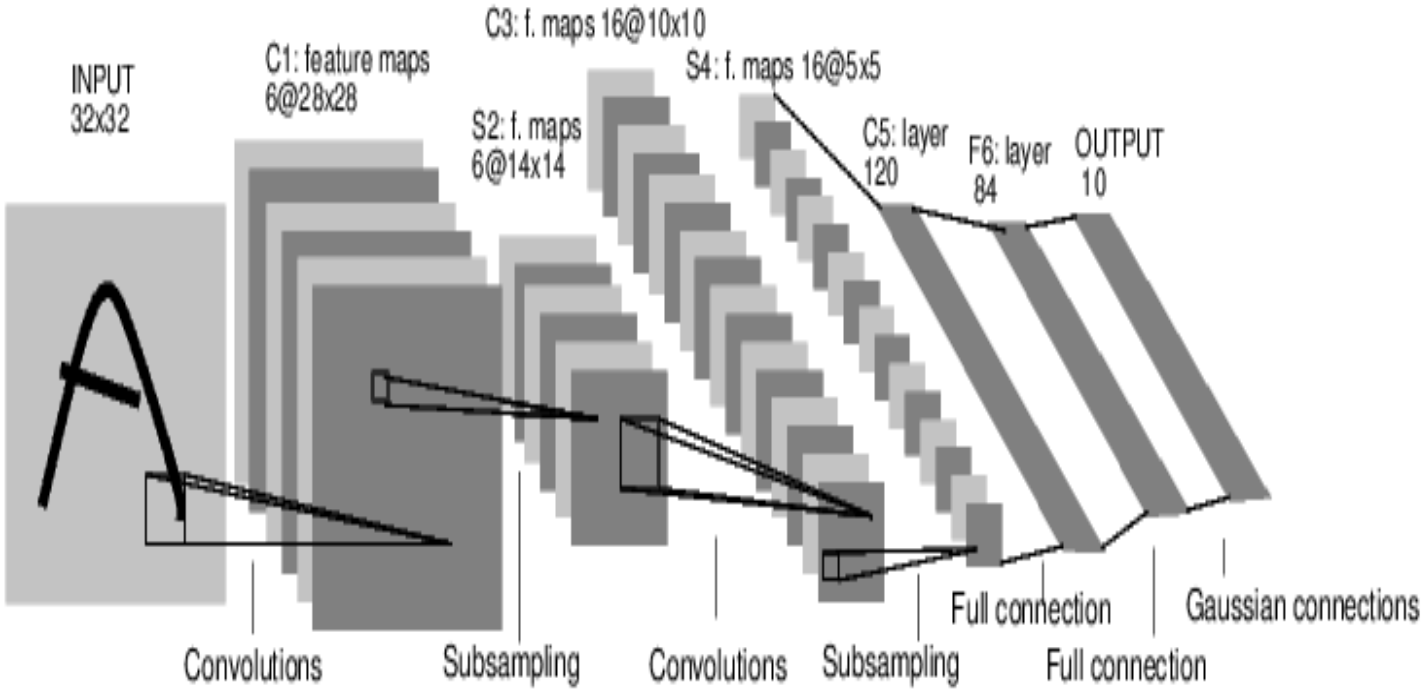


Figure: 卷积操作

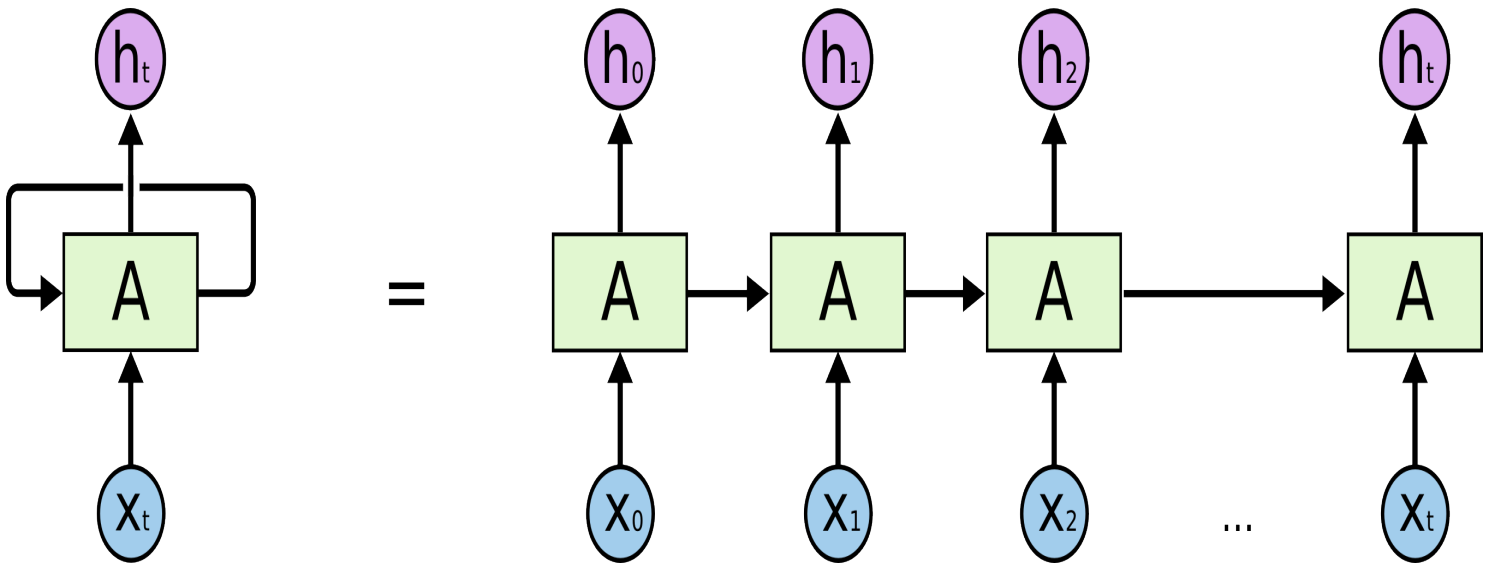
卷积神经网络Convolutional neural network (CNN)

LeCun等人开创性的建立了数字分类的神经网络。几家银行使用它来识别支票上的手写数字。



递归神经网络Recurrent neural networks (RNN)

RNN建立在与前馈神经网络相同的计算单元上。RNN不必分层组织，并且允许定向循环。这样一来，他们就可以拥有内部存储器，从而可以处理顺序数据。如图所示，可以将RNN按顺序“展开”，从而将RNN转换为常规前馈神经网络。



提纲

全局和局部最优解

在求解最优化问题之前，先介绍最小化问题(1) 的最优解的定义.

定义 (最优解)

对于可行点$\bar{x}$ (即$\bar{x} \in \mathcal{X}$)，定义如下概念：

(1) 如果$f(\bar{x}) \leq f(x), \forall x \in \mathcal{X}$，那么称$\bar{x}$ 为问题(1) 的全局极小解（点），有时也称为（全局）最优解或最小值点；

(2) 如果存在$\bar{x}$ 的一个ε 邻域$N_\varepsilon(\bar{x})$ 使得$f(\bar{x}) \leq f(x), \forall x \in N_\varepsilon(\bar{x}) \cap \mathcal{X}$, 那么称$\bar{x}$ 为问题(1) 的局部极小解（点），有时也称为局部最优解；

(3) 进一步地，如果有$f(\bar{x}) < f(x), \forall x \in N_\varepsilon(\bar{x}) \cap \mathcal{X}$, 且$x \neq \bar{x}$ 成立，则称$\bar{x}$ 为问题(1) 的严格局部极小解（点）.

如果一个点是局部极小解，但不是严格局部极小解，则称为非严格局部极小解. 在图4 中，我们以一个简单的函数为例，指出了其全局与局部极小解.

全局和局部最优解

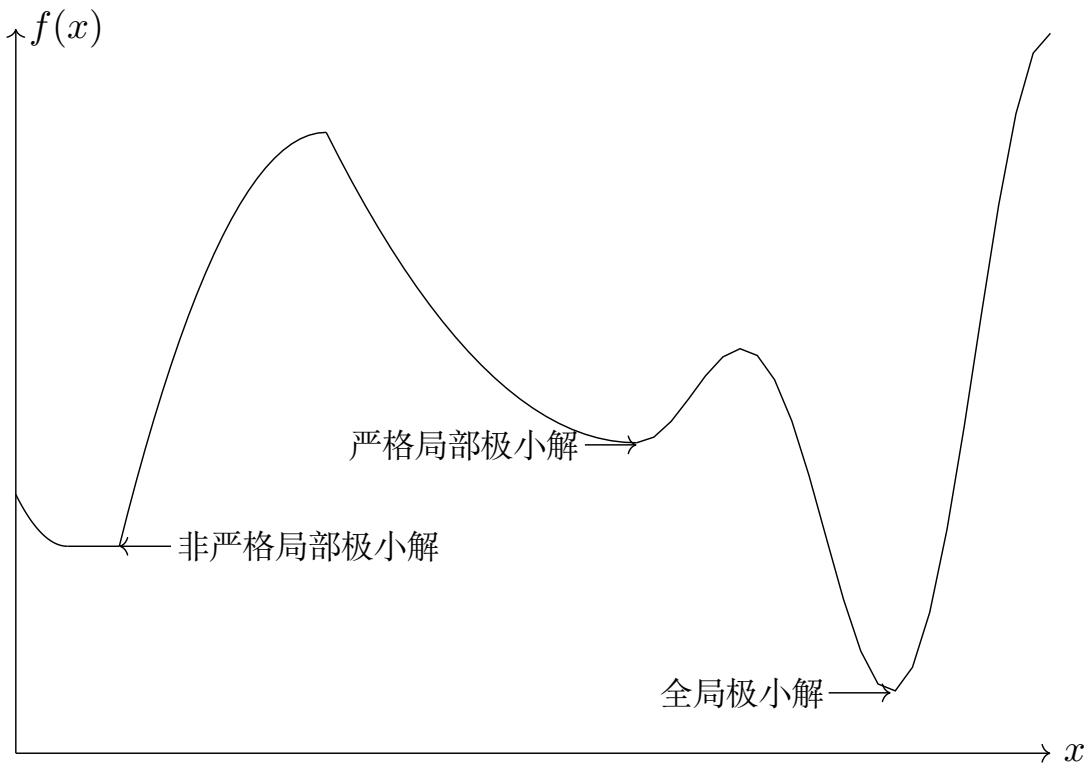


Figure: 函数的全局极小、严格局部极小和非严格局部极小解

在问题(1) 的求解中，我们想要得到的是其全局最优解，但是由于实际问题的复杂性，往往只能够得到其局部最优解.

优化算法收敛性

由于实际问题往往是没有办法显式求解的，因此常采用迭代算法.

- 迭代算法的基本思想是：从一个初始点x^0出发，按照某种给定的规则进行迭代，得到一个序列$\{x^k\}$. 如果迭代在有限步内终止，那么最后一个点就是优化问题的解. 如果迭代点列是无穷集合，那么希望该序列的极限点（或者聚点）则为优化问题的解.
- 在算法设计中，还需考虑算法产生的点列是否收敛到优化问题的解. 给定初始点x^0，记算法迭代产生的点列为$\{x^k\}$. 如果$\{x^k\}$在某种范数$\|\cdot\|$的意义下满足$\lim_{k\to\infty}\|x^k - x^*\| = 0$,且收敛的点$x^*$为一个局部（全局）极小解，那么我们称该点列收敛到局部（全局）极小解，相应的算法称为是依点列收敛到局部（全局）极小解的.

优化算法收敛性

- 如果从任意初始点x^0出发，算法都是依点列收敛到局部（全局）极小解的，我们称该算法是全局依点列收敛到局部（全局）极小解的．记对应的函数值序列$\{f(x^k)\}$，可以定义算法的（全局）依函数值收敛到局部（全局）极小值的概念．
- 对于凸优化问题，因为其任何局部最优解都为全局最优解，算法的收敛性都是相对于其全局极小而言的．
- 除了点列和函数值的收敛外，实际中常用的还有每个迭代点的最优性条件（如无约束优化问题中的梯度范数，约束优化问题中的最优性条件违反度等等）的收敛．
- 对于带约束的情形，给定初始点x^0，算法产生的点列$\{x^k\}$不一定是可行的（即$x^k \in \mathcal{X}$未必对任意k成立）．考虑到约束违反的情形，我们需要保证$\{x^k\}$在收敛到x^*的时候，其违反度是可接受的．除此要求之外，算法的收敛性的定义和无约束情形相同．

算法的渐进收敛速度

设$\{x^k\}$为算法产生的迭代点列且收敛于x^*

- 算法（点列）**Q-**线性收敛：对充分大的k有

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|}\leq a,\quad a\in(0,1)$$

- 算法（点列）**Q-**超线性收敛：对充分大的k有

$$\lim_{k\to\infty}\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|}=0,$$

- 算法（点列）**Q-**次线性收敛：对充分大的k有

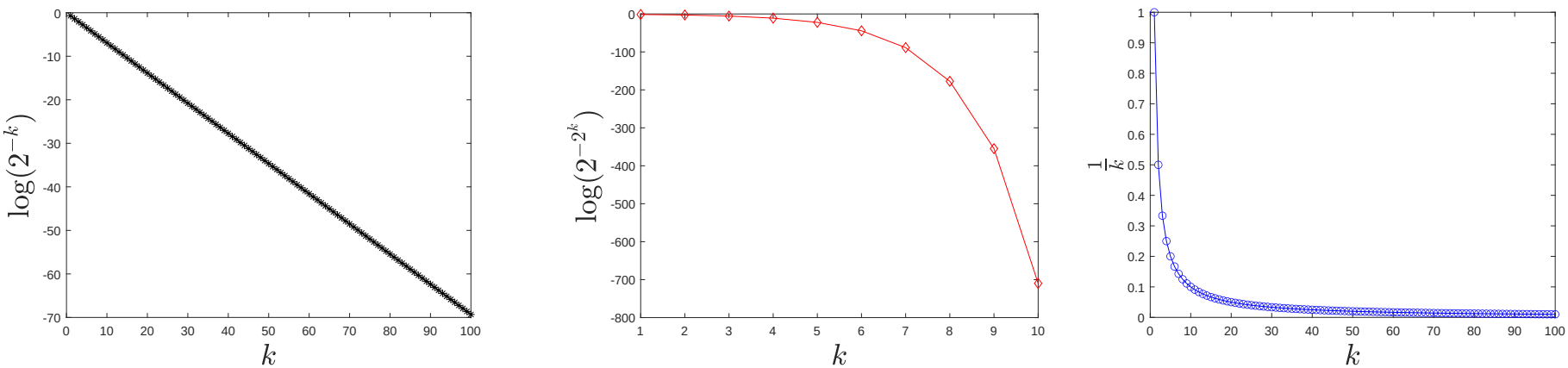
$$\lim_{k\to\infty}\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|}=1,$$

- 算法（点列）**Q-**二次收敛：对充分大的k有

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^2}\leq a,\quad a>0,$$

算法的渐进收敛速度

我们举例来更直观地展示不同的Q-收敛速度，参见下图（图中对所考虑的点列作了适当的变换）．点列$\{2^{-k}\}$是Q-线性收敛的，点列$\{2^{-2^k}\}$是Q-二次收敛的（也是Q-超线性收敛的），点列$\{\frac{1}{k}\}$是Q-次线性收敛的．一般来说，具有Q-超线性收敛速度和Q-二次收敛速度的算法是收敛较快的



(a) Q-线性收敛　(b) Q-二次收敛　(c) Q-次线性收敛

Figure: 不同Q-收敛速度比较

算法的渐进收敛速度

- 算法（点列）**R-**线性收敛: 设$\{x^k\}$为算法产生的迭代点且收敛于x^*，若存在Q-线性收敛于0 的非负序列t_k 并且

$$\|x^k - x^*\| \leq t_k$$

对任意的k成立．类似地，可定义**R-**超线性收敛和**R-**二次收敛等收敛速度．从R-收敛速度的定义可以看出序列$\{\|x^k - x^*\|\}$ 被另一趋于0 的序列$\{t_k\}$控制．当知道t_k的形式时，我们也称算法（点列）的收敛速度为$\mathcal{O}(t_k)$.

- 算法复杂度．设x^*为全局极小点，某一算法产生的迭代序列$\{x^k\}$满足

$$f(x^k) - f(x^*) \leq \frac{c}{\sqrt{k}}, \quad \forall\, k > 0,$$

其中$c > 0$为常数。如果需要计算算法满足精度$f(x^k) - f(x^*) \leq \varepsilon$所需的迭代次数，只需令$\frac{c}{\sqrt{k}} \leq \varepsilon$ 则得到$k \geq \frac{c^2}{\varepsilon^2}$，因此该优化算法对应的（迭代次数）复杂度为$N(\varepsilon) = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$.

优化算法的收敛准则

为了使算法能在有限步内终止，一般会通过一些收敛准则来保证迭代停在问题的一定精度逼近解上.

- 对于无约束优化问题，常用的收敛准则有

$$\frac{f(x^k)-f^*}{\max\{|f^*|,1\}} \le \varepsilon_1, \quad \|\nabla f(x^k)\| \le \varepsilon_2, \tag{7}$$

其中$\varepsilon_1, \varepsilon_2$为给定的很小的正数，$\|\cdot\|$表示某种范数（这里可以简单理解为$\ell_2$范数：$\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$，$f^*$为函数$f$的最小值以及$\nabla f(x^k)$表示函数$f$在点$x$处的梯度.

- 对于约束优化问题，还需要考虑约束违反度. 也即要求最后得到的点满足

$$c_i(x^k) \le \varepsilon_3, \ i = 1, 2, \cdots, m,$$
$$|c_i(x^k)| \le \varepsilon_4, \ i = m+1, m+2, \cdots, m+l,$$

其中$\varepsilon_3, \varepsilon_4$为很小的正数，用来刻画$x^k$的可行性.

优化算法的收敛准则

- 除了约束违反度之外，也要考虑x^k与最优解之间的距离，如(7)式中给出的函数值与最优值的相对误差．由于一般情况下事先并不知道最优解，在最优解唯一的情形下一般使用某种基准算法来得到x^*的一个估计，之后计算其与x^k的距离以评价算法的性能．
- 因为约束的存在，不能简单地用目标函数的梯度来判断最优性，实际中采用的判别准则是点的最优性条件的违反度．
- 对于具体的算法，根据其设计的出发点，不一定能得到一个高精度的逼近解．为了避免无用的计算开销，需要一些停机准则来及时停止算法的进行．常用的停机准则有

$$\frac{\|x^{k+1}-x^k\|}{\max\{\|x^k\|,1\}}\le\varepsilon_5,\qquad \frac{|f(x^{k+1})-f(x^k)|}{\max\{|f(x^k)|,1\}}\le\varepsilon_6,$$

其分别表示相邻迭代点和其对应目标函数值的相对误差很小．

- 在算法设计中，这两个条件往往只能反映迭代点列接近收敛，但不能代表收敛到优化问题的最优解．

提纲

大语言模型辅助工具

大语言模型

- chatgpt: `https://chat.openai.com`
- Claude: `https://claude.ai`
- ...

编程辅助工具

- vscode: Github copilot, copilot chat
- Tabnine
- Cursor

大语言模型辅助推导和写程序

Consider the problem

$$\min_x \mu\|x\|_1 + \|Ax - b\|_1$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are given. Test data are as follows: n = 1024; m = 512; A = randn(m,n); u = sprandn(n,1,0.1); b = A*u; mu = 1e-2;

First we add a slack variable $y = Ax - b$ to the problem, and the problem becomes

$$\min_{x,y} \mu\|x\|_1 + \|y\|_1 \quad \text{s.t. } y = Ax - b$$

and let $z = (x, y)$. Then apply the augmented Lagrangian method to the problem, and the augmented Lagrangian function is

$$L_\rho(z,\lambda) = \mu\|x\|_1 + \|y\|_1 + \lambda^T(Ax - y - b) + \frac{\rho}{2}\|Ax - y - b\|_2^2$$

where λ is the Lagrangian multiplier and ρ is the penalty parameter. The augmented Lagrangian method is

$$\begin{aligned} z^{k+1} &= \arg\min_z L_\rho\left(z, \lambda^k\right) \\ \lambda^{k+1} &= \lambda^k + \rho\left(Ax^{k+1} - y^{k+1} - b\right) \\ k &= k+1 \end{aligned}$$

where x^{k+1} and y^{k+1} are obtained by minimizing $L_\rho\left(z, \lambda^k\right)$ with respect to x and y respectively. Note that minimizing $L_\rho\left(z, \lambda^k\right)$ with respect to x is equivalent to computing a proximal mapping of 1-norm, Please compute it as explicit formula.

Write down and implement the following algorithms in Matlab: Classical Augmented Lagrangian method (or Bregman method), where each augmented Lagrangian function is minimized by using the proximal gradient method Reference: Wotao Yin, Stanley Osher, Donald Goldfarb, Jerome Darbon, Bregman Iterative Algorithms for l1-Minimization with Applications to Compressed Sensing

```
function [x, y, lambda] = AugmentedLagrangian(A, b, mu, rho, max_iter)

    [m, n] = size(A);
    x = zeros(n, 1);
    y = zeros(m, 1);
    lambda = zeros(m, 1);

    for k = 1:max_iter

        % Compute the gradient of the augmented Lagrangian w.r.t. x and y
        grad_x = A'*(lambda + rho*(A*x - y - b)) + mu;
        grad_y = -lambda - rho*(A*x - y - b);

        % Update x and y using the proximal mapping (gradient step + projection)
        x = prox_l1(x - (1/rho)*grad_x, mu/rho);
        y = y - (1/rho)*grad_y;

        % Update the Lagrange multiplier
        lambda = lambda + rho*(A*x - y - b);
    end

end


function w = prox_l1(v, mu)
    w = sign(v).*max(abs(v)-mu,0);
end
```

数学形式化证明：Lean

- A **formal proof** in Lean is a machine-checkable representation of a mathematical argument.
- Utilizes a rigorous foundation based on type theory (依赖类型论).
- Explicitly defines each step, ensuring all logical inferences are justified.
- Leverages Lean's *tactic* language to facilitate proof construction.
- Lean官方网站：
  `https://leanprover-community.github.io/`
- Mathlib is a user maintained library for the Lean theorem prover. It contains both programming infrastructure and mathematics, as well as tactics that use the former and allow to develop the latter.
- Mathlib官方网站：
  `https://github.com/leanprover-community/mathlib4`

Differences from Textbook Proofs

- **Verbosity**: Lean proofs are more verbose and detail-oriented, leaving no room for ambiguity.
- **Interactivity**: Lean proofs are constructed interactively using proof assistants.
- **Machine-Checkable**: Every step in a Lean proof is verified by the system, ensuring correctness.
- **Absence of Prose**: While textbook proofs often include prose explanations, Lean proofs are more symbolic and formal.
- **Modularity**: Lean allows for the reuse of proven lemmas and theorems, modularizing the proof process.

定理陈述: 二次上界

- （二次上界）设可微函数$f(x)$的定义域$\mathbf{dom} f = \mathbb{R}^n$，且为梯度$L$-利普希茨连续的，则函数$f(x)$有二次上界：

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}\|y - x\|^2, \quad \forall\, x, y \in \mathbf{dom} f.$$

- 形式化：
  - 定义域：$\mathbb{R}^n \to$ EuclideanSpace $\mathbb{R}$ n
  - $\nabla f(x) \to$ HasFDerivAt f (f' x) x
    注意f' 的类型
    (EuclideanSpace $\mathbb{R}$ n) $\to$ ((EuclideanSpace $\mathbb{R}$ n) $\to$L[$\mathbb{R}$] $\mathbb{R}$)
  - 梯度L-利普希茨连续$\to \|f'(x) - f'(y)\| \leq L\|x - y\|$
  - 二次上界$\to f(y) \leq f(x) + f'(x)(y - x) + L\|y - x\|^2/2$

Frechet导数的mathlib定义

- 大括号：定义类
- 中括号：定义代数结构
- 小括号：需要给出的条件

```
def HasFDerivAt                                                    source
       {𝕜 : Type u_1} [inst : NontriviallyNormedField 𝕜] {E : Type u_2}
       [inst : NormedAddCommGroup E] [inst : NormedSpace 𝕜 E]
       {F : Type u_3} [inst : NormedAddCommGroup F]
       [inst : NormedSpace 𝕜 F] (f : E → F) (f' : E →L[𝕜] F) (x : E)
       :
   Prop
```

A function `f` has the continuous linear map `f'` as derivative at `x` if `f x' = f x + f' (x' - x) + o (x' - x)` when `x'` tends to `x`.

▶ Equations
▶ Instances For

Figure: Frechet导数的mathlib定义

定理陈述的形式化

Mathlib 库+ 变量+ 定理陈述

```
import Mathlib.Tactic
import Mathlib.Analysis.Calculus.Deriv.Add
import Mathlib.Analysis.Calculus.Deriv.Comp
import Mathlib.Analysis.Calculus.MeanValue
import Mathlib.Analysis.Calculus.Deriv.Basic
import Mathlib.Analysis.InnerProductSpace.PiL2
import Mathlib.Analysis.Calculus.FDeriv.Basic

variable {n : Type _}[Fintype n]{f : (EuclideanSpace ℝ n) → ℝ}
{f' : ((EuclideanSpace ℝ n)) → ((EuclideanSpace ℝ n) →L[ℝ] ℝ)} {l : ℝ}

theorem lipschitz_continuos_upper_bound
(h₁ : ∀ x₁ :EuclideanSpace ℝ n, HasFDerivAt f (f' x₁) x₁)
(h₂ : ∀ x y : EuclideanSpace ℝ n, ‖f' x - f' y‖ ≤ l * ‖x - y‖):
∀ (x y : EuclideanSpace ℝ n), f y ≤ f x + (f' x) (y - x) + l / 2 * ‖y - x‖ ^ 2 := by
```

Figure: 定理陈述的形式化

定理陈述的形式化

- 在定理陈述中挑选出关键的数学对象。
- 阅读mathlib库并寻找数学对象对应的文档。
  - 如果你能找到：检查定义是否与你期望的定义方式相同、学习使用方式。
  - 如果不能：给出你自己的定义。
- 将自然语言转换成Lean语言：注意那些隐藏在自然语言中的信息。
  - 处理“常识”问题。
  - 不要向定理中添加额外的信息。
- 列出你需要的所有变量和假设，然后形成定理。
- 对于引入库：在开始时引入一些基本的库，需要时补充相应的库。

证明的形式化

- 令$g(t)=f(x+t(y-x))$，可以得到$g'(t)=\nabla f(x+t(y-x))^T(y-x)$。

```
intro x y
let g := fun t : ℝ ↦ f (x + t • (y - x))
let h := fun t : ℝ ↦ x + t • (y - x)
have h_deriv : ∀ t : ℝ,  HasDerivAt h (y - x) t := by
  intro t
  rw [hasDerivAt_iff_isLittleO_nhds_zero]
  have :(fun h => (t + h) • (y - x) - t • (y - x) - h • (y - x)) = (fun h => 0) := by
    ext h
    rw [add_smul, add_comm, ← add_sub, sub_self, add_zero, sub_self]
  simp [this]
have deriv_comp: ∀ t₀ : ℝ , HasDerivAt (f ∘ h) (f' (x + t₀ • (y - x)) (y - x)) t₀ := by
  intro t₀
  apply HasFDerivAt.comp_hasDerivAt
  · apply h₁ (x + t₀ • (y - x))
  · apply h_deriv t₀
let g' := fun t : ℝ ↦ (f' (x + t • (y - x)) (y - x))
have deriv_comp' : ∀ t₀ : ℝ , HasDerivAt  g (g' t₀) t₀ := fun t₀ ↦ deriv_comp t₀
```

Figure: $g(t)$的梯度

Gpt4的能力

目标: 证明$g'(t) = \nabla f(x + t(y - x))^T (y - x)$
我们将这个要求输入到Gpt4。结果如下：

```
import analysis.calculus.deriv
import analysis.calculus.fderiv
import analysis.normed_space.inner_product
variables {n : Type _} [fintype n]
variables {f : EuclideanSpace ℝ n → ℝ}
variables {f' : EuclideanSpace ℝ n → (EuclideanSpace ℝ n →L[ℝ] ℝ)}
variables {l : ℝ} {x y : EuclideanSpace ℝ n}
hypothesis h₁ : ∀ x₁ : EuclideanSpace ℝ n, has_fderiv_at f (f' x₁) x₁
def g := λ t : ℝ, f (x + t • (y - x))
lemma has_deriv_at_g (t₀ : ℝ) : has_deriv_at g (f' (x + t₀ • (y - x)) (y - x)) t₀ :=
begin
  have hf' : has_fderiv_at f (f' (x + t₀ • (y - x))) (x + t₀ • (y - x)),
  { exact h₁ (x + t₀ • (y - x)) },
  rw [g],
  exact hf'.has_deriv_at_linear_map (has_deriv_at_id t₀).smul_const (y - x),
end
```

Figure: gpt4的结果

Gpt4的能力

一个良好的形式化草稿。

- 优点
  - 结构正确：库+ 变量+ 假设+ 定理+ 证明。
  - 思想基本正确：提到了相关的主要定理。
  - 能将大目标分解成小目标。
- 不足
  - 语法存在问题：更高效的prompt engineering
  - 定理的名称需要更新：人类可以帮助从Lean3转换到Lean4。
  - 只有简短的证明（对于长证明gpt不会写出具体代码）。
- 结论
  - 可以提示需要使用的定理。
  - 可以生成框架性结构，人类专家再做填充。

总结

在数学中形式化某个定理：

- 形式化定理陈述，包括条件和结果：
  - 为特定的数学对象选择特定的形式化方式。
  - 提供所需的变量。
  - 形式化所有的条件。
- 使用自然语言形成证明草稿。
- 选择重要的中间引理并将整个证明划分为几个部分。
- 在mathlib中找到适当的定理以帮助我们形式化证明。