IFI 9000 Analytics Methods
Convex Optimization

by **Houping Xiao**

Spring 2021

Introduction

Mathematical Optimization

- **(Mathematical) optimization problem**

$$\begin{array}{ll} \underset{\boldsymbol{\beta}}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq b_i, \forall i = 1, \cdots, m \end{array}$$

- $x = (x_1, \cdots, x_n)$: optimization variables
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function
- $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \cdots, m$: constraint functions

- **optimal solution** x^* has smallest value of f among all vectors that satisfy the constraints

Examples

- **portfolio optimization**
  - variables: amounts invested in different assets
  - objective: overall risk or return variance
  - constraints: budget, max./min. investment per asset, minimum return
- **data fitting**
  - variables: model parameters
  - objective: measure of misfit or prediction error
  - constraints: prior information, parameter limits

Solving optimization problems

- Usually, it's very difficult to solve the **general optimization problem**
- The methods involve some compromise, e.g., very long computation time, or not always finding the solution

- There are some **exceptions** that certain problem classes can be solved efficiently and reliably
  - least-squares problems
  - linear programming problems
  - convex optimization problems

Least-squares problems

- **Least-squares problems :** Optimize the square loss (distance) without constraints

$$\underset{x}{\text{minimize}} \quad ||Ax - b||_2^2$$

- **solutions**
  - The optimal(analytical) solution is that $x^* = (A^\top A)^{-1} A^\top b$
  - There are reliable and efficient algorithms and software, such as `lm` in R and `scipy.optimize` in Python
  - The computation time of solving the least-squares problems is proportional to $n^2 k$ given $A \in \mathbb{R}^{k \times n}$; less if structured (i.e., $\boldsymbol{x}$ is sparse)

- **using least-squares**
  - Least-squares problems are easy to recognize
  - There are a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

Linear Programming

- **Linear Programming**: Optimize a linear function subject to linear inequalities.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

- **solutions**:
  - no analytical formula, but there are reliable and efficient algorithms and software
  - The computation time of solving the linear programs is proportional to $n^2 m$ if $m > n$; less with structure

- **using linear programming**
  - not as easy to recognize as least-squares problems
  - there are a few standard tricks used to convert problems into linear programs. For instance, problems involving l_1-norms, piecewise-linear functions

Convex optimization problem

- The formula with a **convex optimization** is that

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq b_i, \forall i = 1, \cdots, k \end{aligned}$$

  where both objective and constraint function are convex functions:

$$g_i(\alpha x + \beta y) \leq \alpha g_i(x) + \beta g_i(y)$$

  if $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

- The convex optimization includes least-square problems and linear programs as special cases

Solving convex optimization problems

- Usually, there is no analytical solution, but with reliable and efficient algorithms
- The computation time proportional to $\max\{n^3, n^2m, F\}$ where F is cost of evaluating f and g_i and their first and second derivatives

using convex optimization

- Sometimes, it's often difficult to recognize
- There are many tricks for transforming problems into convex form. Surprisingly many problems can be solved via convex optimization

Solving an optimization: a general perspective

- Consider an unconstrained, smooth convex optimization

$$\min_{x} \quad f(x)$$

  - f is convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$
  - optimal criterion value $f^* = \min_{x} \quad f(x)$
  - a optimal solution x^*

- A necessary and sufficient condition for a point x^* to be optimal is

$$\bigtriangledown f(x^*) = 0$$

  - $\bigtriangledown f(x)$ is easy to obtain
  - But, $\bigtriangledown f(x)$ doesn't have a straightforward solution?
  - **(Batch) Descent Methods: Gradient Descent, Stochastic Gradient Descent, etc**

Descent Methods

- Consider an unconstrained, smooth convex optimization

$$\underset{x}{min} \quad f(x)$$

- Find a sequence: $x^{(0)}, x^{(1)}, \cdots, \in \mathrm{dom}(f)$, s.t.

$$\lim_{k\to\infty} f(x^{(k)}) \to f^*$$

- descent methods:

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \quad s.t. \quad f(x^{(k+1)}) < f(x^{(k)})$$

- gradient descent: Initialize $x^{(0)}$, repeat:

$$x^{(k+1)} = x^{(k)} - t_k \dot{\bigtriangledown} f(x^{(k)}), \quad k = 1, 2, 3, \cdots$$

Stop at some point (i.e., x no change!)

Gradient Descent Methods

"Gradient descent is a **first-order** iterative optimization algorithm for finding the minimum of a function."

- for each k, based on the Taylor theorem

$$f(y) \approx f(x) + \triangledown f(x)^\top (y-x) + \frac{1}{2}(y-x) \triangledown^2 f(x)(y-x)$$

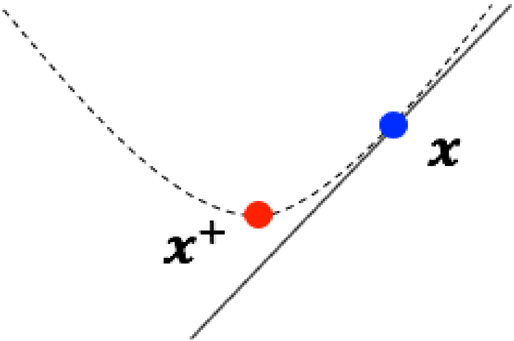
  - quadratic approximation: replace Hessian matrix $\triangledown^2 f$ by $\frac{1}{t} I$

$$f(y) \approx f(x) + \triangledown f(x)^\top (y-x) + \frac{1}{2t}||y-x||_2^2$$

  - linear approximation to f, proximity term to x, with weight $\frac{1}{2t}$
- choose next point $y = x^+$ to minimize quadratic approximation:

$$x^+ = x - t \triangledown f(x)$$

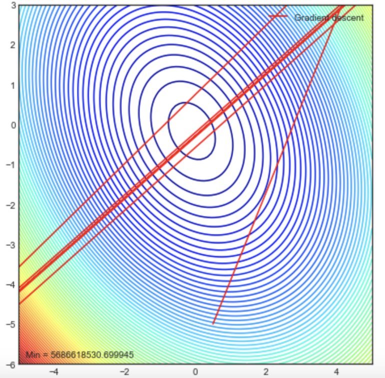
Gradient Descent Methods



$$x^+ = \arg\min_y \quad f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2t} ||y - x||_2^2$$

How to choose step size or learning rate t?

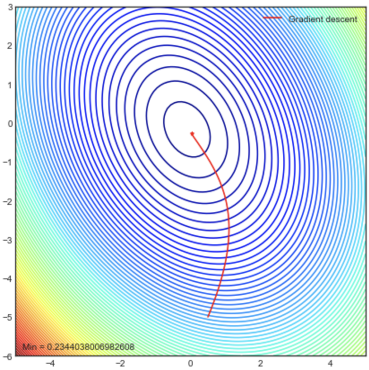
- **Fixed step size strategy**: at each step, the step size or learning rate t_k is fixed, i.e., $t_k = t$ for all $k = 1, 2, 3, \cdots$,
- **Issues :** can diverge if t is too big



Large step size: 10 iterations

How to choose step size or learning rate t?

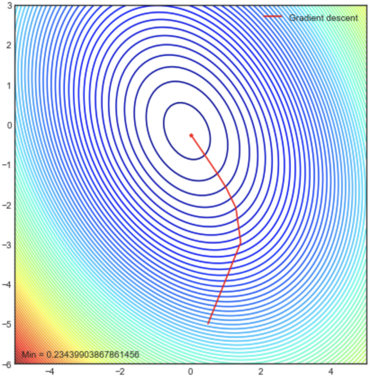
- **Fixed step size strategy**: at each step, the step size or learning rate t_k is fixed, i.e., $t_k = t$ for all $k = 1, 2, 3, \cdots$,
- **Issues :** can converge super slow if t is too small



Small step size: 1000 iterations

How to choose step size or learning rate t?

- **Fixed step size strategy**: at each step, the step size or learning rate t_k is fixed, i.e., $t_k = t$ for all $k = 1, 2, 3, \cdots$,
- **Issues :** can converge fast if t is been carefully chosen



"Just right" step size: 40 iterations

Backtracking line search: Adaptively choose step size

- **backtracking line search** is one way to adaptively choose the step size

---

Algorithm 1: Gradient descent with Backtracking line search

$\alpha \in (0, 0.5), \beta \in (0, 1)$;
given a starting point $x \in \mathrm{dom}(f)$;
initialization, set $t = t^0$;
repeat
    determine a descent direction $\bigtriangledown f(x)$;
    while $f(x - t \bigtriangledown f(x)) > f(x) - \alpha || \bigtriangledown f(x)||_2^2$ **do**
        set $t = \beta \cdot t$;
    end
    update $x = x - t \bigtriangledown f(x)$;
until *stopping criterion is satisfied*;

- simple and tends to work well in practice (further simplification: $\alpha = 0.5$)

Backtracking (line search) Interpretation

for us
$\Delta x = -\bigtriangledown f(x)$

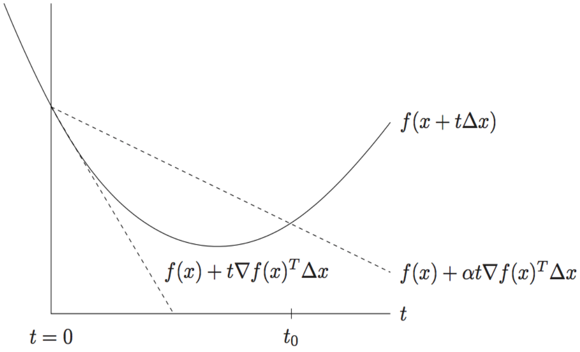


Figure 9.1 *Backtracking line search.* The curve shows f, restricted to the line over which we search. The lower dashed line shows the linear extrapolation of f, and the upper dashed line has a slope a factor of α smaller. The backtracking condition is that f lies below the upper dashed line, *i.e.*, $0 \leq t \leq t_0$.

Exact line search: select the best step size

- Exact line search is able to choose optimal step size along direction of negative gradient

$$t = \arg\min_{s \geq 0} \quad f(x - s \bigtriangledown f(x))$$

  - Usually not possible to exactly minimize $f(x - s \bigtriangledown f(x))$
  - Approximations to Exact line search are typically not as efficient as backtracking (not worth it!)

Convergence analysis

- Given f convex and differentiable, with $\mathrm{dom}(f) = \mathbb{R}^n$, and $\bigtriangledown f$ is Lipschitz continuous with constant $L > 0$,

$$||\bigtriangledown f(x) - \bigtriangledown f(y)||_2 \leq L||x - y||_2, \text{for any} x, y$$

Theorem

Gradient descent with fixed step size $t \leq \frac{1}{L}$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{||x^{(0)} - x^*||_2^2}{2tk}$$

and same results holds for backtracking, with $t = \frac{\beta}{L}$.

- Gradient descent has convergence rate $\mathcal{O}(1/k)$, i.e., it takes $\mathcal{O}(1/\epsilon)$ itesration for gradient descent to find a ϵ-suboptimal point.

Convergence analysis: Analysis for strong convexity

- strong convexity: $f(x) - \frac{m}{2}||x||_2^2$ is convex for some $m > 0$

Theorem

Given that f strong convex, Lipschitz continuous, gradient descent with fixed step size $t \leq \frac{2}{m+L}$ or with backtracking line search satisfies

$$f(x^{(k)}) - f^* \leq \gamma^k \frac{L}{2}||x^{(0)} - x^*||_2^2$$

where $0 < \gamma < 1$

- convergence rate is $\mathcal{O}(\gamma^k)$, exponentially fast! Now, it takes only $\mathcal{O}(\log(1/\epsilon))$ to find a ϵ-suboptimal point.

Exact line search v.s. backtracking line search

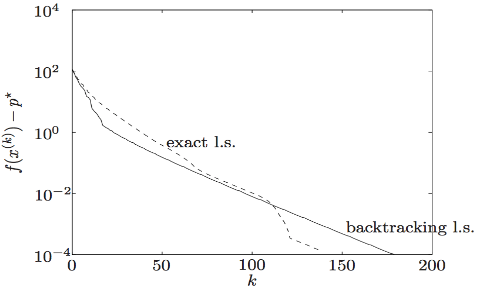


Figure 9.6 Error $f(x^{(k)}) - p^\star$ versus iteration k for the gradient method with backtracking and exact line search, for a problem in $\mathbf{R}^{100}$.

- $\gamma = \mathcal{O}(1 - m/L)$, the convergence rate reduces to

$$\mathcal{O}(\frac{L}{m} \log(1/\epsilon))$$

- higher condition number $L/m \to$ slower rate
  - not only true in theory, but also apparent in practice

An example of checking the conditions

- goal:

$$f(\beta) = \frac{1}{2}||y - X^\top \beta||_2^2$$

- Lipschitz continuity of $\bigtriangledown f$:
  - recall this means $\bigtriangledown^2 f(x) \preceq LI$
  - $\bigtriangledown^2 f(\beta) = X^\top X \rightarrow L = \lambda_{max}(X^\top X)$
- Strong convexity of f:
  - $\bigtriangledown^2 f(x) \succeq mI$
  - $\bigtriangledown^2 f(\beta) = X^\top X \rightarrow m = \lambda_{min}(X^\top X)$

Practicality tricks

- stopping rule: stop when $|| \bigtriangledown f(x)||_2$ is small
  - recall $\bigtriangledown f(x^*) = 0$ at solution x^*
  - if f is strongly convex with m, then

$$|| \bigtriangledown f(x)||_2 \leq \sqrt{2m\epsilon} \Rightarrow f(x) - f^* \leq \epsilon$$

- Pros and cons
  - pros:
    - simple idea, and each iteration is cheap
    - fast for well-conditioned, strongly convex problems
  - cons:
    - can often be slow, because many of none convexity or not well-conditioned
    - can't handle non-differential functions
    - Non-convex optimization!

Stochastic gradient descent

- consider minimizing an average of functions

$$\min_x \quad \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

- gradient descent:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \boldsymbol{\frac{1}{m}\sum_{i=1}^{m}} \nabla \boldsymbol{f_i(x^{(k-1)})}, k = 1, 2, 3, \cdots,$$

- stochastic gradient descent (SGD) repeats:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla \boldsymbol{f_{i_k}(x^{(k-1)})}, k = 1, 2, 3, \cdots,$$

where index $i_k \in \{1, \cdots, m\}$ is chosen at iteration k

How to choose index i_k

- Randomly or cyclically select sample gradient:
  - randomized rule: choose $i_k \in \{1, \cdots, m\}$ uniformly at random
    - more common in practice
    - $\mathbb{E}(\bigtriangledown f_{i_k}(x)) = \bigtriangledown f(x)$
    - an unbiased estimate of gradient at each step
  - cyclic rule choose $i_k = 1, 2, \cdots, m, 1, 2, \cdots, m, \cdots$
- main appeal of SGD:
  - The iteration cost is independent of number of functions
  - SGD will save big a lot in memory usage, compared with batch GD

An example of SGD: stochastic logistic regression

$$\min_{\beta} \quad \frac{1}{m}\sum_{i=1}^{m} \underbrace{\left(-y_i x_i^\top \beta + \log(1+\exp(x_i^\top \beta))\right)}_{f_i(\beta)}$$

where $(x_i, y_i) \in \mathbb{R}^n \times \{0,1\}, i = 1, 2, \cdots, n$

- $\bigtriangledown f(\beta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - p_i(\beta))x_i$
- full gradient (i.e. batch) v.s. stochastic gradient:
  - one batch update costs $\mathcal{O}(np)$
  - one stochastic update costs $\mathcal{O}(p)$
- if large amount of steps are needed, SGD is much more affordable

How to choose step size?

- diminishing step sizes: $t + k = \frac{1}{k}$
- why not fixed step size?
  - use cyclic rule
  - $t_k = t$ for m updates in a row, we have

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^{m} \bigtriangledown f_i(x^{(k+i-1)})$$

  - batch gradient with step size mt is:

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^{m} \bigtriangledown f_i(x^{(k)})$$

  - difference:

$$\Delta = t \sum_{i=1}^{m} [\bigtriangledown f_i * x^{(k+i-1)} - \bigtriangledown f_i(x^{(k)})]$$

  if t is constant, Δ won't go to zero

Convergence rates for SGD

- for convex f, SGD with diminishing step size satifies

$$\mathbb{E}(f(x^{(k)}) - f^* = \mathcal{O}(1/\sqrt{k})$$

  - stays the same even if f is Lipschitz gradient
- for strongly convex, SGD has

$$\mathbb{E}(f(x^{(k)})) - f^* = \mathcal{O}(1/k)$$

so, stochastic methods do not enjoy the linear convergence rate of gradient descent under strong convexity

Improve SGD using mini-batches

- mini-batch stochastic gradient descent: randomly choose a subset $I_k \subseteq \{1, \cdots, m\}$, with $|I_k| << m$, do:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{b} \sum_{i \in I_k} \bigtriangledown f_i(x^{(k-1)}), k = 1, 2, 3, \cdots$$

- approximate full gradient by an unbiased estimate:

$$\mathbb{E}\left(\frac{1}{b} \sum_{i \in I_k} \bigtriangledown f_i(x^{(k-1)})\right) = \bigtriangledown f(x)$$

  - reduces variance by a $\frac{1}{b}$
  - b times more expensive in computation

An example of SGD: logistic regression

$$\min_{\beta} \quad \frac{1}{m}\sum_{i=1}^{m}\left(-y_i x_i^\top \beta + \log(1+\exp(x_i^\top \beta))\right) + \frac{\lambda}{2}||\beta||_2^2$$

where $f_i(\beta) = -y_i x_i^\top \beta + \log(1+\exp(x_i^\top \beta)) + \frac{\lambda}{2}||\beta||_2^2$

- gradient : $\bigtriangledown f(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i(\beta))x_i + \lambda\beta$
- update costs
  - one batch: $\mathcal{O}(np)$
  - one mini-batch: $\mathcal{O}(bp)$
  - one stochastic: $\mathcal{O}(p)$

An example of SGD: logistic regression

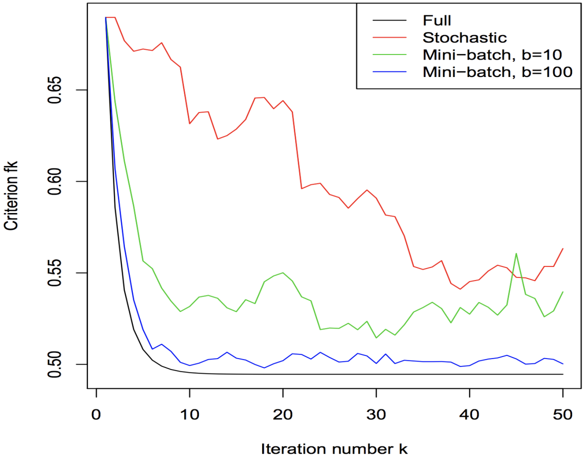


Figure: Example with $n = 10,000, p = 20$, all methods use fixed step size

Early stopping

- for the regularized logistic regression:

$$\min_{\beta} \quad \frac{1}{m}\sum_{i=1}^{m}\left(-y_i x_i^\top \beta + \log(1+\exp(x_i^\top \beta))\right), \quad \text{s.t.} \quad ||\beta||_2^2 \le t$$

- we could also use early stopping to run gradient descent on the unregularized problem:

$$\min_{\beta} \quad \frac{1}{m}\sum_{i=1}^{m}\left(-y_i x_i^\top \beta + \log(1+\exp(x_i^\top \beta))\right)$$

Early stopping

- early stopping:
  - start with $\beta^{(0)}$, solution to regularized problem at $t = 0$
  - run gradient descent on unregularized criterion:

$$\beta^{(k)} = \beta^{(k-1)} - \epsilon \cdot \frac{1}{n} \sum_{i=1}^{n} (y_i - p_i(\beta^{(k-1)}))x_i, k = 1, 2, 3, \cdots$$

  - treat $\beta^{(k)}$ is an spproximate solution to regularized problem with $t = ||\beta^{(k)}||_2$
  - why early stopping?
    - more convenient
    - efficient than using explicit regularization

Concludes of SGD

- SGD can be super effective w.r.t. iteration cost, memory
- SGD is slow to converge, not for strong convexity
- in many ml problems we are not caring about optimizing to high accuracy
- fixed step sizes commonly used
- conduct experiments on a small fraction
- momentum/acceleration, averaging,adaptive step sizes are all popular variants in practice
- SGD is popular in large-scale, continuous, non-convex optimization

Lagrangian

- What if we have constraints in the optimization problems?

$$\begin{aligned} \underset{\beta}{\textit{minimize}} \quad & f(\beta) \\ \text{subject to} \quad & g_i(\beta) \leq 0, \forall i = 1, \cdots, k \\ & h_j(\beta) = 0, \forall j = 1, \cdots, l \end{aligned} \tag{1}$$

variable β, domain $\mathcal{D}$, optimal value p^*

- **Lagrangian:**

$$\mathcal{L}(\beta, \alpha_i, \gamma_j) = f(\beta) + \sum_{i=1}^{k} \alpha_i g_i(\beta) + \sum_{j=1}^{l} \gamma_j h_j(\beta)$$

  - weighted sum of objective and constraint functions
  - α_i is Lagrange multiplier associated with $g_i(\beta) \leq 0$
  - γ_j is Lagrange multiplier associated with $h_j(\beta) = 0$

Langrange dual function

- **Lagrange dual function** g

$$
\begin{aligned}
g(\alpha,\gamma) &= \inf_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta},\alpha_i,\gamma_j) \\
&= \inf_{\boldsymbol{\beta}} \left(f(\boldsymbol{\beta}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{\beta}) + \sum_{j=1}^{l} \gamma_j h_j(\boldsymbol{\beta}) \right)
\end{aligned}
$$

- **lower bound property:** if $\alpha > 0$, then $g(\alpha,\gamma) \leq p^*$
- **weak duality:** $d^* \leq p^*$
- **strong duality:** $d^* = p^*$ (usually holds for convex problems)
- **Karush-Kuhn-Tucker (KKT) conditions:**
  - primal constraints: $g_i(\beta) \leq 0$, $h_j(\beta) = 0$
  - dual constraints: $\alpha \geq 0$
  - complementary slackness $\alpha_i g_i(\beta) = 0$
  - gradient of Lagrangian w.r.t. β vanishes

Linear Programming

- **Linear Programming**: Optimize a linear function subject to linear inequalities.

$$\begin{array}{lll} \max & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j = b_i, & 1 \le i \le m \\ & x_j \ge 0, & 1 \le j \le n \end{array} \qquad \begin{array}{ll} \max & c^\top x \\ \text{s.t.} & Ax = b \\ & x \ge 0 \end{array}$$

- **Generalizes**: 2-person zero-sum games, shortest path, max flow, assignment problem, matching ...

A Toy Example of Linear Programming

Brewery Problem

- Small Brewery produces two products: ale and beer
  - production is limited by scarce resources: corn, hops, barley malt
  - recipes for ale and beer require different proportions of resources:

Beverage	Corn (pounds)	Hops (ounces)	Malt (pounds)	Profit (Dollar)
Ale (barrel)	5	4	35	13
Beer (barrel)	15	4	20	13
Constraints	480	160	1190	

- How to maximize profits?
  - 34 barrels of ale: 442$?
  - 32 barrels of beer: 736$?
  - 7.5 barrels of ale, 29.5 barrels of beer: 776$?
  - 12 barrels of ale, 28 barrels of beer: 800$?

A Toy Example of Linear Programming

Brewery Problem

- Small Brewery produces two products: ale and beer
  - production is limited by scarce resources: corn, hops, barley malt
  - recipes for ale and beer require different proportions of resources:

Beverage	Corn (pounds)	Hops (ounces)	Malt (pounds)	Profit (Dollar)
Ale (barrel)	5	4	35	13
Beer (barrel)	15	4	20	13
Constraints	480	160	1190	

- Objective function, constraints and decision variables X, Y

$$\begin{aligned} \text{maximize} \quad & 13X + 23Y \\ \text{s.t.} \quad & 5X + 15Y \leq 480 \\ & 4X + 4Y \leq 160 \\ & 35X + 20Y \leq 1190 \\ & X, Y \geq 0 \end{aligned}$$

Standard form of a linear programming

- Let's check the standard form of an LP problem
  - **input:** real numbers a_{ij}, c_j and b_i
  - **output:** real numbers x_j
  - n : decision variables; m : constraints number
  - **objective:** maximize (or minimize) linear objective function subject to linear inequalities
    - that means NO x^2, xy, $arccos(x)$, etc

$$\begin{array}{lll} \max & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j = b_i, & 1 \leq i \leq m \\ & x_j \geq 0, & 1 \leq j \leq n \end{array} \qquad \begin{array}{ll} \max & c^\top x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}$$

Some tricks to equivalent forms transformation of the functions

- by introducing a nonnegative slack variable s, a less inequality constraint can be reduced to an equality constraint:

  $$x + 2y - 3z \leq 17 \Rightarrow x + 2y - 3z + s = 17, s \geq 0$$

- similarly, a greater inequality can also be transformed to an equality constraint:

  $$x + 2y - 3z \geq 17 \Rightarrow x + 2y - 3z - s = 17, s \geq 0$$

  s is a nonnegative slack variable

- the minimize objective function can be changed to a maximize objective function:

  $$\min\ (x + 2y - 3z) \Rightarrow \max\ (-x - 2y + 3z)$$

- the unrestricted constraint is equivalent to two nonnegative conditions:

  $$x \text{ unrestricted} \Rightarrow x = x^+ - x^-, x^+ \geq 0, x^- \geq 0$$

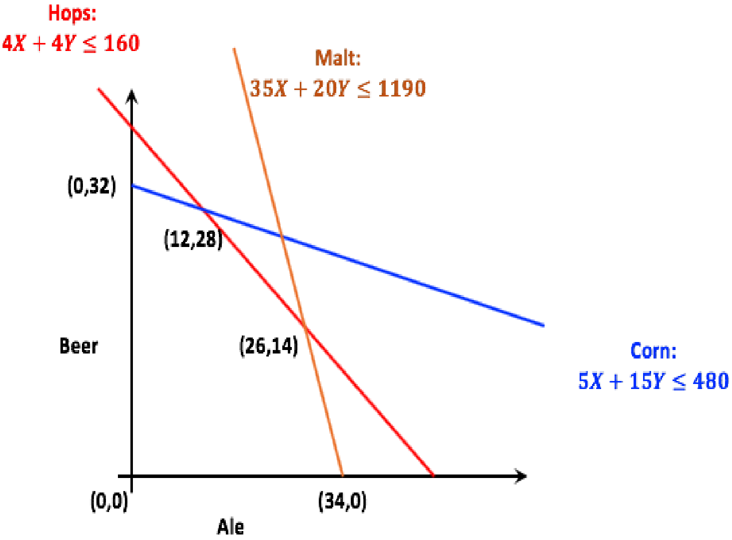
Converting Brewery problem to a standard form

$$\begin{aligned} \max \quad & 13X + 23Y \\ \text{s.t.} \quad & 5X + 15Y \leq 480 \\ & 4X + 4Y \leq 160 \\ & 35X + 20Y \leq 1190 \\ & X, Y \geq 0 \end{aligned}$$

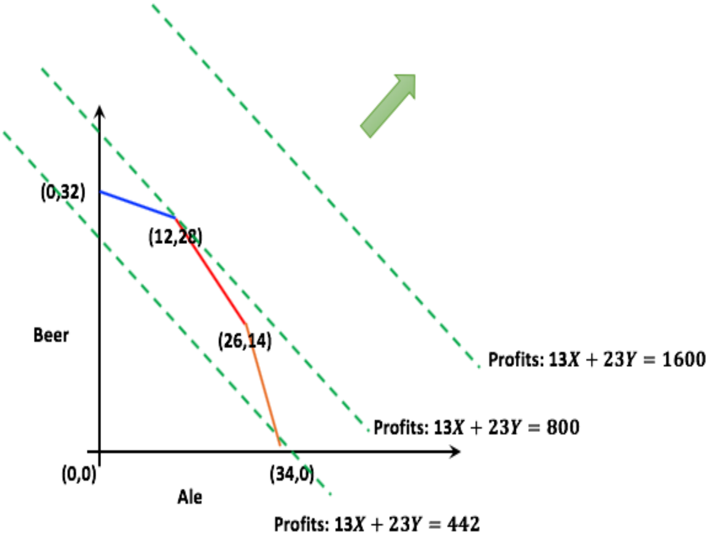
$$\begin{aligned} \max \quad & 13X + 23Y \\ \text{s.t.} \quad & 5X + 15Y + S_A = 480 \\ & 4X + 4Y + +S_B = 160 \\ & 35X + 20Y + S_C = 1190 \\ & X, Y, S_A, S_B, S_C \geq 0 \end{aligned}$$

- Here, we introduce the Non-negative Slack variables: S_A, S_B, S_C

Brewery problem: feasible region

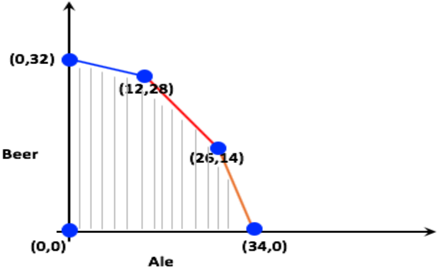


Brewery problem: objective function



Brewery problem: geometry

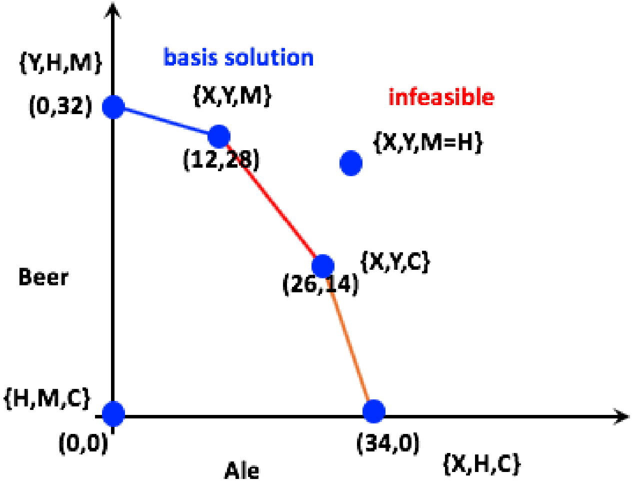
- Brewery problem observation.
  - regardless of objective function coefficients, an optimal solution occurs at a vertex



- convex set: if two points x and y are in the set, then so is $\lambda x + (1 - \lambda) y$ for any $\lambda \in [0, 1]$
- vertex: a point x in the set that can not be written as a strict convex combination of two distinct points in the set

Basis feasible solution: example

- Basis feasible solutions



Linear programming duality

- primal problem

$$
\begin{aligned}
(P) \quad \max \quad & 13X + 23Y \\
\text{s.t.} \quad & 5X + 15Y \leq 480 \\
& 4X + 4Y \leq 160 \\
& 35X + 20Y \leq 1190 \\
& X, Y \geq 0
\end{aligned}
\tag{2}
$$

- Goal:
  - find a lower bound on optimal value
  - find an upper bound on optimal value

Linear programming duality

- primal problem

$$(P) \quad \begin{array}{ll} \max & 13X + 23Y \\ \text{s.t.} & 5X + 15Y \leq 480 \\ & 4X + 4Y \leq 160 \\ & 35X + 20Y \leq 1190 \\ & X, Y \geq 0 \end{array} \tag{3}$$

- Idea: add non-negative combination (C, H, M) of constraints s.t.

$$\begin{aligned} 13X + 23Y &\leq (5C + 4H + 35M) \cdot X + (15C + 4H + 20M) \cdot Y \\ &\leq 480C + 160H + 1190M \end{aligned}$$

- dual problem: find best such upper bound

$$(D) \quad \begin{array}{ll} \min & 480C + 160H + 1190M \\ \text{s.t.} & 5C + 4H + 35M \geq 13 \\ & 15C + 4H + 35M \leq 23 \\ & C, H, M \geq 0 \end{array}$$

Linear programming duality

economic interpretation

- Brewer to find optimal mix of bear and ale to maximize profits

$$
\begin{aligned}
(P) \quad \max \quad & 13X + 23Y \\
\text{s.t.} \quad & 5X + 15Y \leq 480 \\
& 4X + 4Y \leq 160 \\
& 35X + 20Y \leq 1190 \\
& X, Y \geq 0
\end{aligned} \tag{4}
$$

- Entrepreneur to buy individual resources from brewer at min cost
  - C, H, M =unit price for corn, hops malt
  - Brewer won't agree to see resources if "$5C + 4H + 35M < 13$"

$$
\begin{aligned}
(P) \quad \min \quad & 480C + 160H + 1190M \\
\text{s.t.} \quad & 5C + 4H + 35M \geq 13 \\
& 15C + 4H + 20M \geq 23 \\
& C, H, M \geq 0
\end{aligned} \tag{5}
$$

- canonical form

$$(P)\quad \max \; c^\top x \quad \text{s.t.} \; Ax \le b,\; x \ge 0 \qquad\qquad (D)\quad \min \; y^\top b \quad \text{s.t.} \; A^\top y \ge c,\; y \ge 0$$

- property: the dual of the dual is the primal

Primal (P)	Maximize
constraints	$ax = b_i$ $ax \le b_i$ $ax \ge b_i$
variables	$x_j \ge 0$ $x_j \le 0$ x_j unrestricted

Minimize	Dual (D)
y_i unrestricted $y_i \ge 0$ $y_i \le 0$	variables
$a^\top y \ge c_j$ $a^\top y \le c_j$ $a^\top y = c_j$	constraints

Linear programming strong and weak duality

LP strong duality

for $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$, if (P) and (D) are nonempty, then max $=$ min

$$(P) \quad \begin{array}{ll} \max & c^\top x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array} \qquad\qquad (D) \quad \begin{array}{ll} \min & y^\top b \\ \text{s.t.} & A^\top y \geq c \\ & y \geq 0 \end{array}$$

LP weak duality

for $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$, if (P) and (D) are nonempty, then max $\leq$ min

$$(P) \quad \begin{array}{ll} \max & c^\top x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array} \qquad\qquad (D) \quad \begin{array}{ll} \min & y^\top b \\ \text{s.t.} & A^\top y \geq c \\ & y \geq 0 \end{array}$$

Linear programming duality: sensitivity analysis

- How much should brewer be willing to pay (marginal price) for additional supplies of scarce resources?
  - corn \$ 1, hops \$ 2, malt \$0
- Suppose a new product “light beer” is proposed. It requires 2 corn, 5 hops, 24 malt. How much profit must be obtained from light beer to justify diverting resources from production of beer and ale?
  - At least 2 (\$1) + 5 (\$2) + 24 (\$0) = \$12 / barrel.

The End