**Coursera – Discrete Optimization**

**LP 1: Intuition, Convexity, Geometric View (**[**link**](https://www.coursera.org/learn/discrete-optimization/lecture/UU9w9/lp-1-intuition-convexity-geometric-view)**)**

This lecture covers three topics:

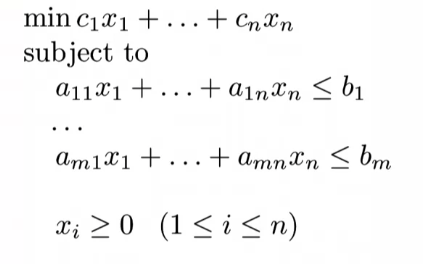
* What is Linear Programming (LP)?
* Convexity and its importance in LP
* Geometric representation of LP problems

***What is Linear Programming?***

Invented by George Dantzig in 1947, LP is one of the fundamental tools in *combinatorial optimization* (the topic of finding the optimal solution from a set of feasible solutions which is too large to exhaustively search via brute force).

A nice characteristic of LP is that all solutions derived from LP can be represented both geometrically and algebraically. Typically, when understanding an LP solution, we’ll go back and forth between the algebraic and geometric representations of the problem. Geometric representations allow us to visualize the problem, while algebraic representations allows us to distill the problem into an equation.

A linear program can be generalized as such:

The top equation represents the linear *objective function* which needs to be minimized/optimized. This objective function is subject to a set of linear inequalities called *constraints*. A few characteristics of the equations above:

* All variables need to be non-negative ()
* All of the in the equation above are all variables represent the real numbers included in the problem/dataset
* All of , , and are constants of the algebraic equations included in the problem (note that and are coefficients)
* Using the representation above, there are variables and constraints

The characteristics above naturally lead to these questions about LP:

* Can I maximize the objective function?

Yes. To do this, we can negate the objective function, and minimize this new equation –

* What should I do if a variable can take negative values?

You can replace with two non-negative values, whose difference can be negative. Replacing with *everywhere* in your linear program can enable negative values for your variables.

* Instead of an inequality constraint, what if I have an equality constraint?

This is an easy fix; we can just provide two inequality constraints – one which is the constant, another which is the constant, resulting in an equality constraint.

* What if my variable only takes integer values?

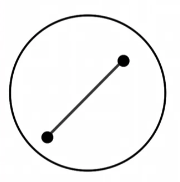
This can’t be handled directly by LP. This is another domain of optimization called *mixed integer linear programming* which is covered in other parts of the course.

* What if I have a non-linear constraint?

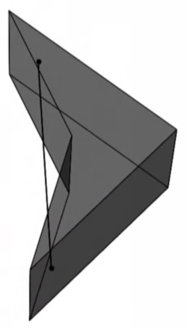
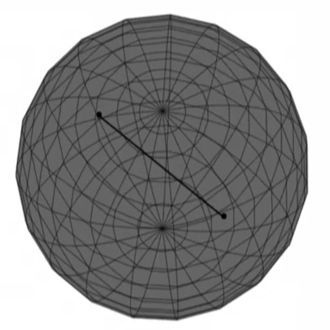
This also is not handled in LP. It is called *linear* programming after all ☺

***Convexity and its importance in LP***(6:00 in video)

Convex sets are a set of points that, given any two points, the line joining them lies entirely within the set. In a 2D space, a circle is a convex set while a star is not.



This can be extended to more dimensions, 3D for example. A sphere is a convex set while the *polyhedron* on the right is not.



The above looks at convex sets graphically, but how we would define a convex combination algebraically is below.

A set of points is a convex combination of if:

and

Put into words, a convex set is any group of points that can multiplied by a set of lambdas. The lambdas are all positive numbers, and the sum of all lambdas is 1.

Another way to describe whether a set is convex or not is to take any two points within the set. If you connect each point to every other point within the set, and those combinations are *all* convex, then the set is convex as well.

Expanding this out from only one convex set to many convex sets, we can look at the intersection of a number of convex sets, denoted by . If we take the points contained in the intersection, it follows that the set of points within are also in each one of the original sets. Furthermore this intersection is also convex.

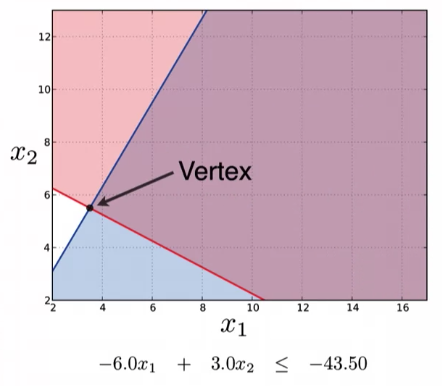
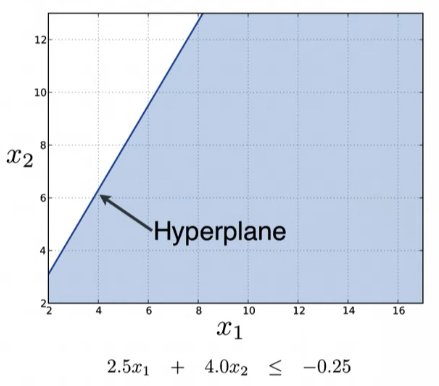
*Why is convexity important in LP?*

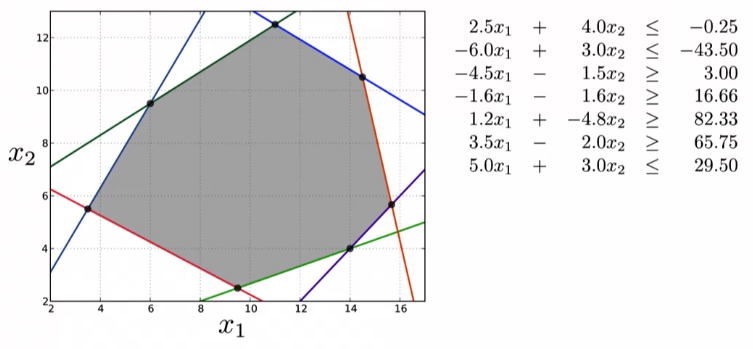
As mentioned, all linear programs include constraints which are inequalities. These inequalities are *half spaces* that divide the dimensional space into two separate convex sets. Within a linear program, you have a bunch of convex half spaces that are defining the feasible region of the program.

In other words, the intersection of all the convex sets defined by the constraints of the LP – called a *polyhedron ­­­*– is the feasible region for the linear program.

In a 2D space, one of the constraints of the LP (defining a single half space) can be represented geometrically through a hyperplane.

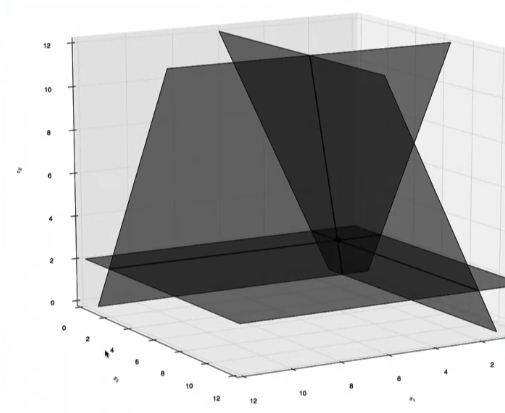
When we add another half space, this refines the feasible region that satisfy all constraints. Additionally, the intersection of the half spaces form a *vertex* which is the point where the hyperplanes intersect.

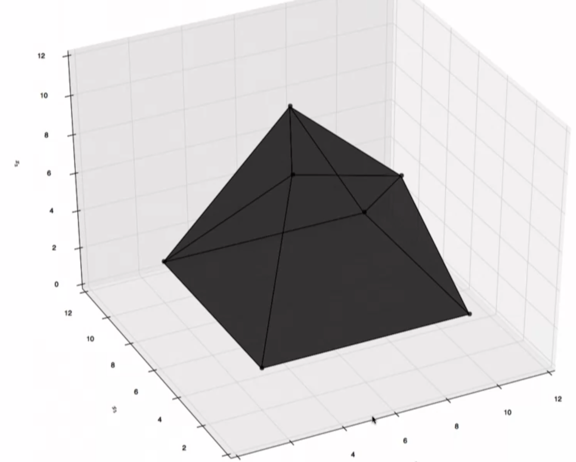


Adding more half space will define a feasible region called a *polytope* since the feasible region does not include infinity.

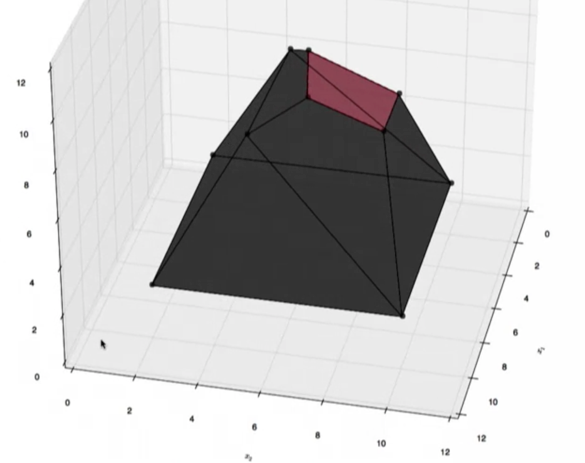
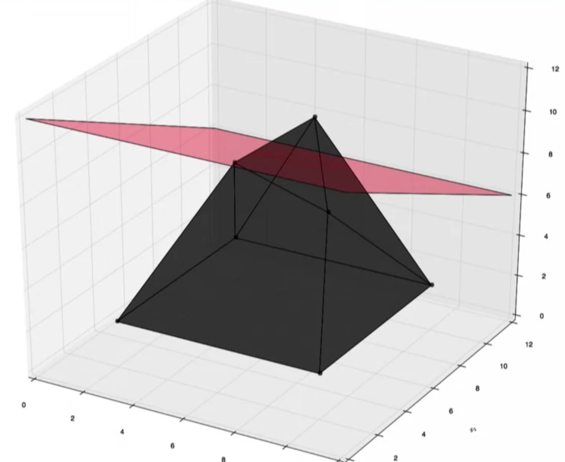
A few important term definitions:

* A *face* is the intersection of a finite number of hyperplanes
* For variables, the point at which hyperplanes intersect is called a *vertex*, while the face created by a hyperplane on an existing polytope is called a *facet*.

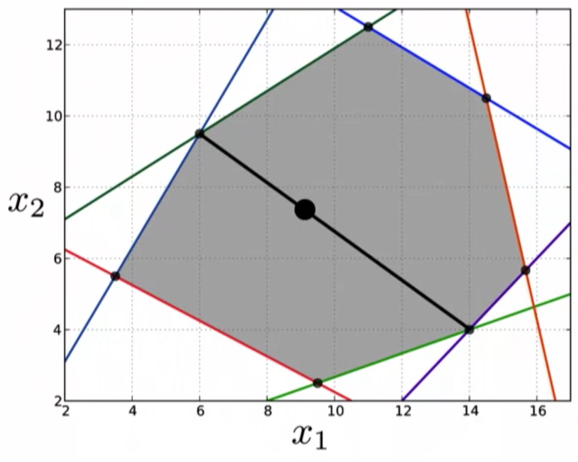
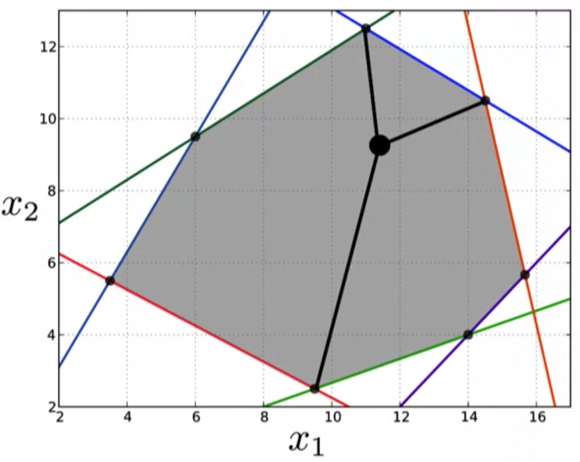
The pictures above depict hyperplanes in a 2D space, below depicts a similar concept in a 3D space. The intersection of three hyperplanes (toward the back of the graph) is the vertex; note that the intersection of two hyperplanes is a line.

Adding more inequalities results in a polytope, an example of which is below. These are the kinds of objects that we are going to explore inside a linear program.

When we add another hyperplane, depicted in red in the graph below, the intersection with the existing polytope creates a few new vertices, as well as a facet of the newly-created polytope.

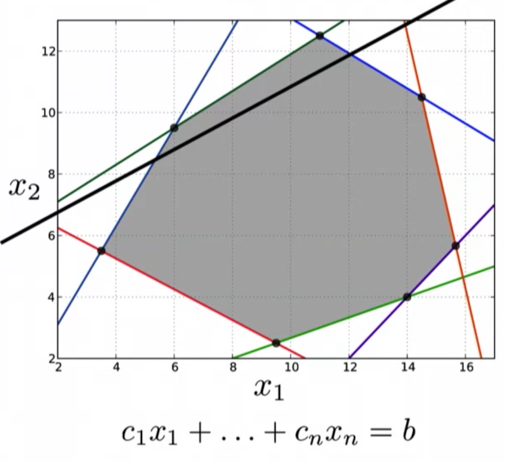


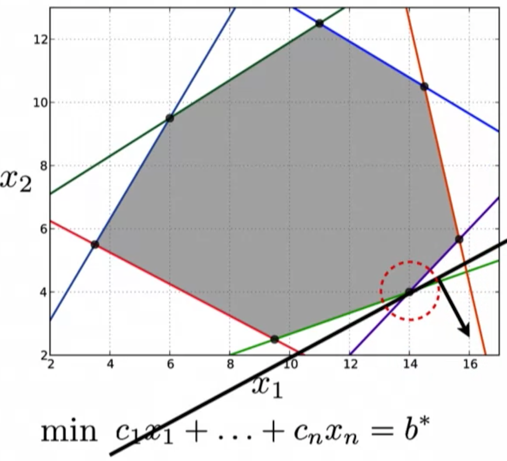
The concept of vertices are extremely important in a linear program. The reason is because any point in a polytope – and subsequently, any feasible solution to a linear program – is a convex combination of its vertices. Sometimes it is only a combination of two vertices, other times it can be three.



Additionally, as we have explained, the point of a linear program is to find the minimum of an objective function.

*At least one of the points where the objective value is minimal is a vertex.*

In a 2D space, imagine that the objective function is the black line below through the polytope defined by our constraints.

We want to find the minimal feasible value that satisfies the objective function, so what we could is push the objective down to its minimum where at least one point of the objective intersects with the polytope of feasible solutions. Notice in the equation at the bottom of the graph, the right-hand side is marked as , which indicates the minimal value of the objective function.

Again extending the example to 3D, the red hyperplane represents the objective function while the polytope is the convex set of feasible solutions; notice that the object function is maximized at a vertex of the polytope.

Because the optimal solution to a linear program is on one of the vertices, we can solve a linear program geometrically by enumerating all the vertices, and selecting the one with the smallest objective value.