Single-Agent Mechanism Design

Michael Curry

CMSC498T

4/25/22

Illustration of problem

- Suppose you're running a fruit stand on the side of the road, selling apples and oranges.
- You want to put a sign up advertising your prices
- People will drive by. If they like your prices, they'll buy, otherwise they will keep driving.
- What prices maximize expected revenue?

Key aspects of this problem

- For 1 good, just set a price (Econ 101 MR = MC)
- For multiple goods, many more decisions
  - Discount for buying in bulk?
  - Discount for bundles?
- Interesting history
  - early work by Adams and Yellen 1976
  - some progress in 2000s (Manelli-Vincent, Pavlov, Daskalakis et al.)
  - still not completely solved
- We usually call the set of prices a “menu”

Alternative – direct revelation mechanism

- You are going to ask a person how much they like apples and oranges.
- Based on what they say, you are going to take some money out of their bank account and give them apples and oranges.
- You don't want people to lie
- It's just single-agent mechanism design.

Review: Mechanism Design Setting

- Private-value model – assume agent has private type x
  - Concretely: it is a vector saying how much each item is worth
- Mechanism asks for x, chooses allocation a(x) and payment p(x)
- Agents can lie about x, and we want to disincentivize this (**strategyproof**), and promise positive utility (**individually rational**)
- We don’t know x, but we know it was drawn from a distribution P(x)
- Goal: maximize expected revenue over P(x)

These are equivalent problems!

- Recall the *revelation principle*: it tells us that menus can be transformed into truthful direct-revelation mechanisms
- Less obviously, for single agents, every truthful direct revelation mechanism has a corresponding menu – this is known as the *taxation principle*
- In fact, there is a relationship of **duality** between direct-revelation mechanisms and menus.

The Outline From Here

- Motivation for problem: picking the optimal menu
- Direct-revelation mechanisms
  - Mechanisms as utility functions
  - Properties of mechanisms ⇔ properties of their utility functions
- Detour: convex functions
  - Convex sets and functions
  - Convex conjugate operation
  - Max-over-affine representation
- Back to menus: conjugates of utility functions
- Wednesday: review, examples of mechanisms, and ML for learning optimal mechanisms

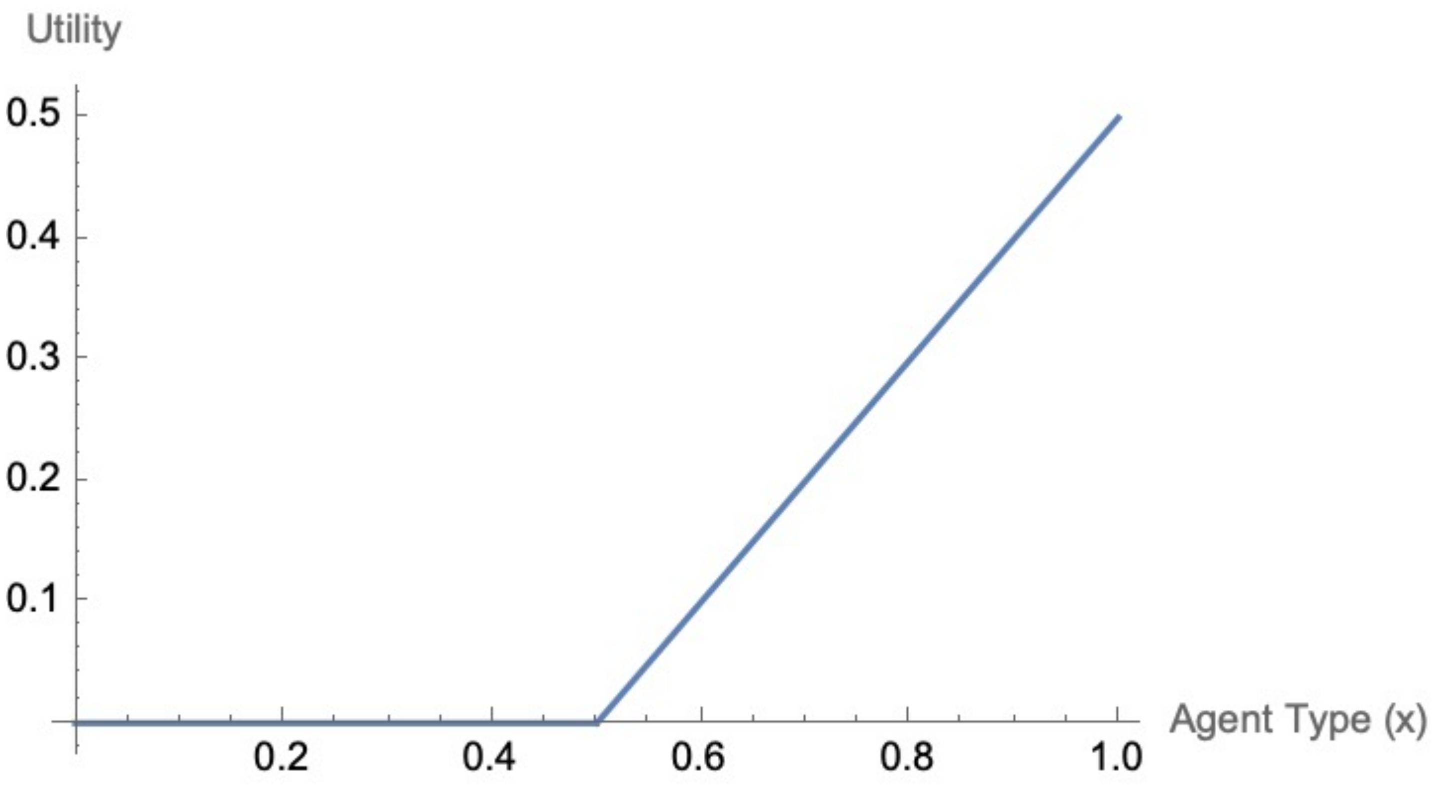
Mechanisms as Utility Functions

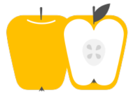
- The **utility function** maps a type (how much each good is valued) to how much utility a bidder with that type will get.
- Two components:
  - You get positive utility for the items you receive
  - You get negative utility for paying money
- Suppose we have any allocation and payment rule – these *induce* some utility function

Formal Definitions

- Private types x denoted in red. Think of it as a column vector giving the true value of each item.
- Allocations a denoted in blue. Think of them as row vectors specifying how much of each item.
- Inner product $a \cdot x$ gives value of an allocation. (Primal/dual relationship)
- Mechanism has allocation rule a(x). Value from allocation is $a(x) \cdot x$ Subtract payment p(x).
- Total utility is $a(x) \cdot x$ – p(x).

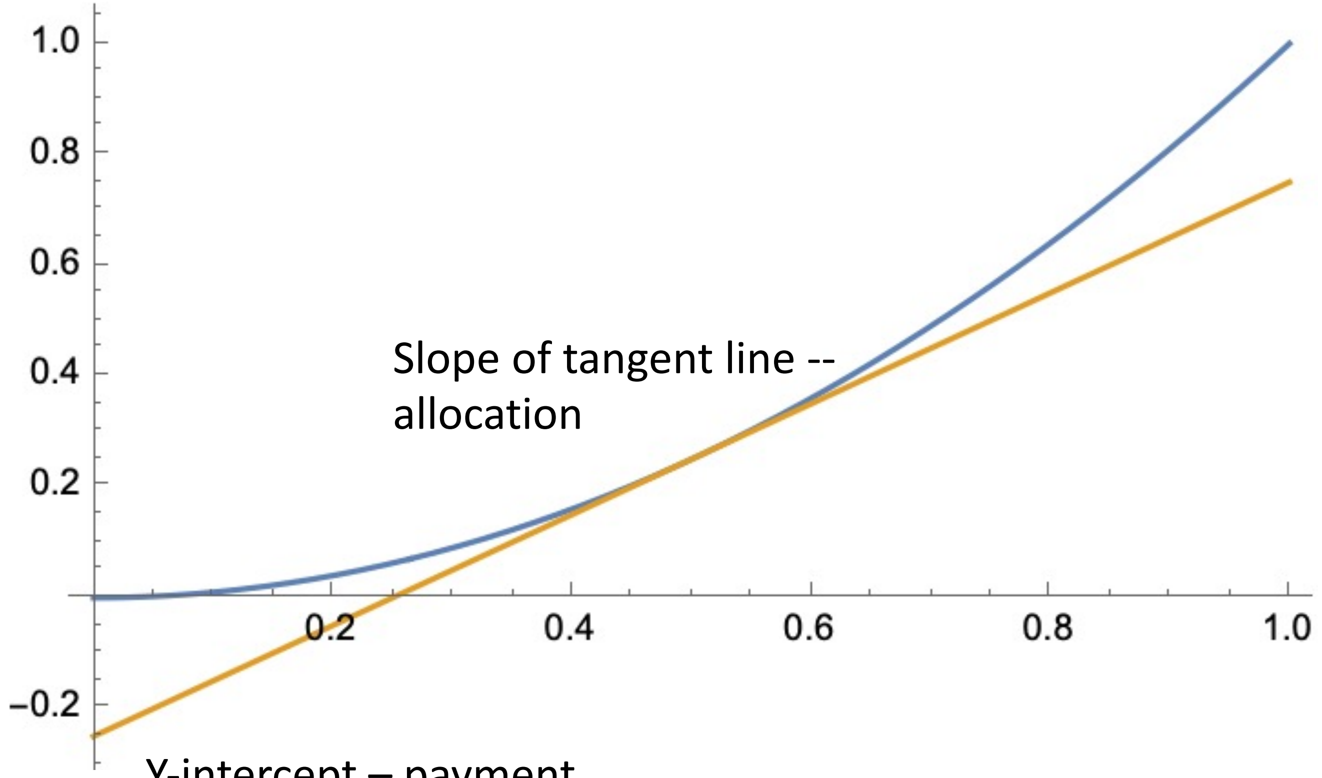
Concrete Example – 1D





$0.50

Every Valid Utility Function Gives Some a, p



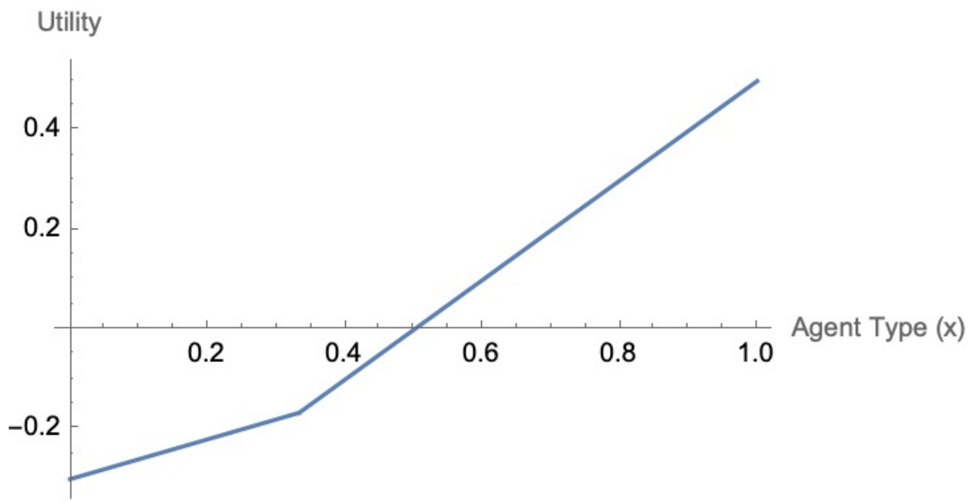
- Mechanism gives utility u(x)
- Assert a(x) = u'(x) – allocation is gradient of utility
- Agent welfare from getting items is u'(x) · x
- Payment p(x) is then necessarily u'(x) · x - u(x)
- u'(x) · x – (u'(x) · x - u(x)) = u(x), so this doesn't cause problems (not a proof though)

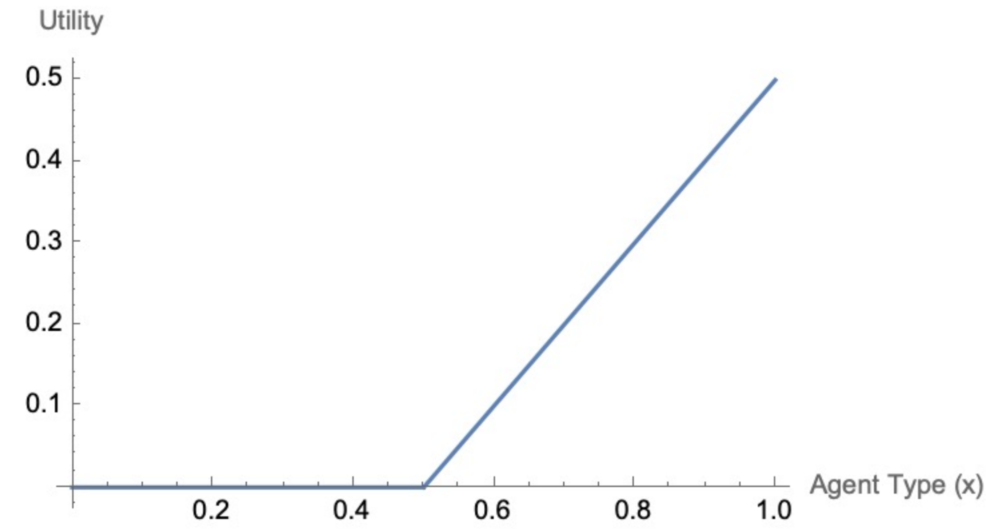
Properties of Utility Functions

- Utility functions can be **identified** with mechanisms
- Recall our mechanism design goal: truthfulness (aka DSIC, strategyproofness) and IR
  - Bidders have some value x but they are allowed to lie about it
  - We want **no incentive to lie**
  - We want **no negative utility**
- What does this mean in terms of the utility function?

Individual Rationality

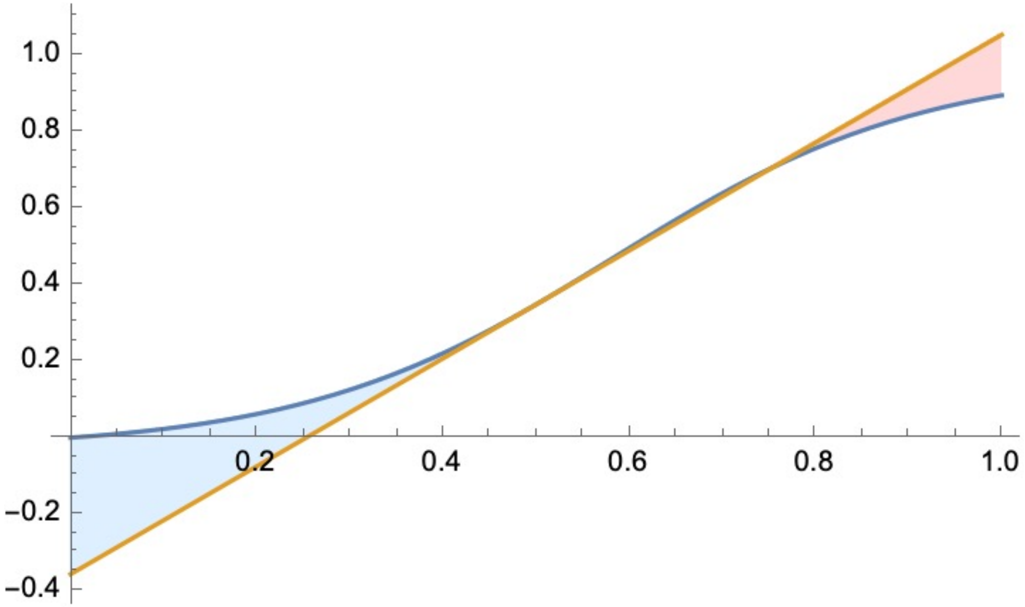
- Individual rationality just means no negative utility
- Literally – just make sure utility function is not negative
- Intuitively, the buyer can always walk away

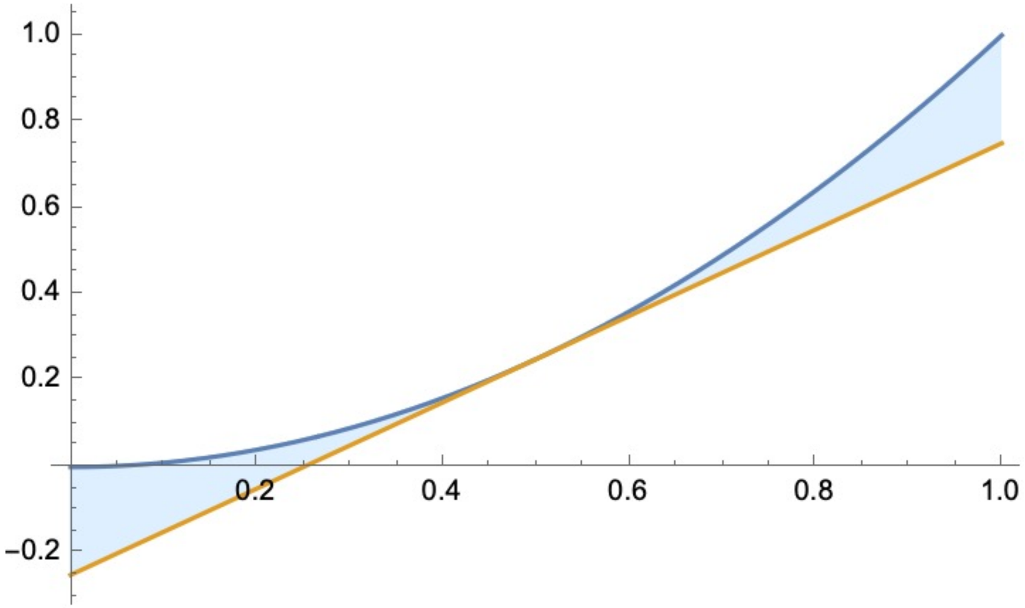




Truthfulness and Convexity

- Blue line: mechanism utility u(x)
- Bid is 0.5. Orange line: utility from untruthfully bidding 0.5, as your true type varies. Tangent to curve (allocation is u'(x))
- If u(x) is non-convex, there is a region (red) where you can benefit from lying





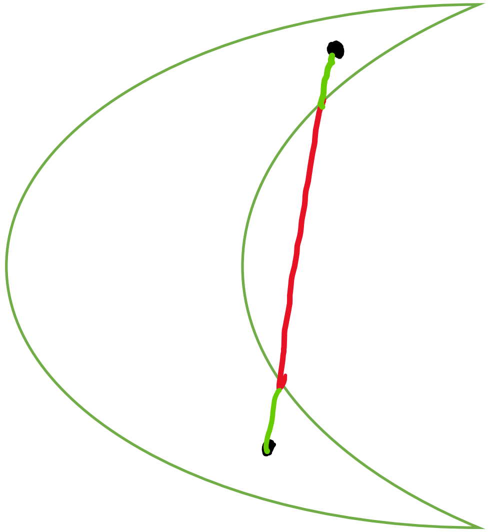
Recap

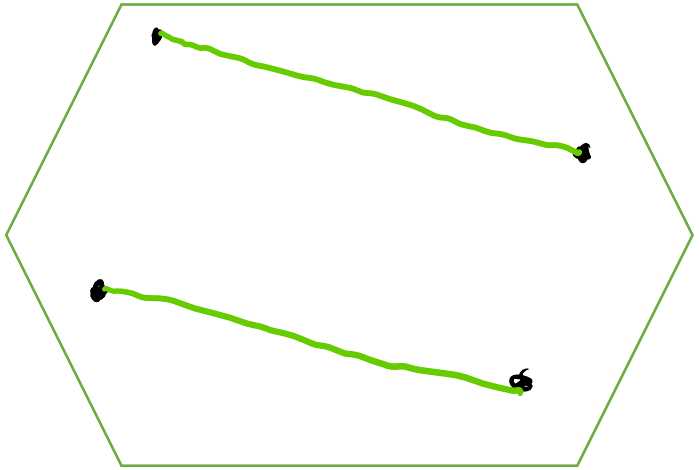
- **Identify** direct revelation mechanisms with utility functions
- Properties we want as mechanism designers
  - **IR** – just make sure function is nonnegative
  - **Strategyproof** – make sure function is convex
- Mechanism design is just choosing your favorite convex, nonnegative utility function (how to do that?)
- Next: more detail on convexity

Convex Sets

- What makes a set convex? ”Draw a line and it stays in the set.”

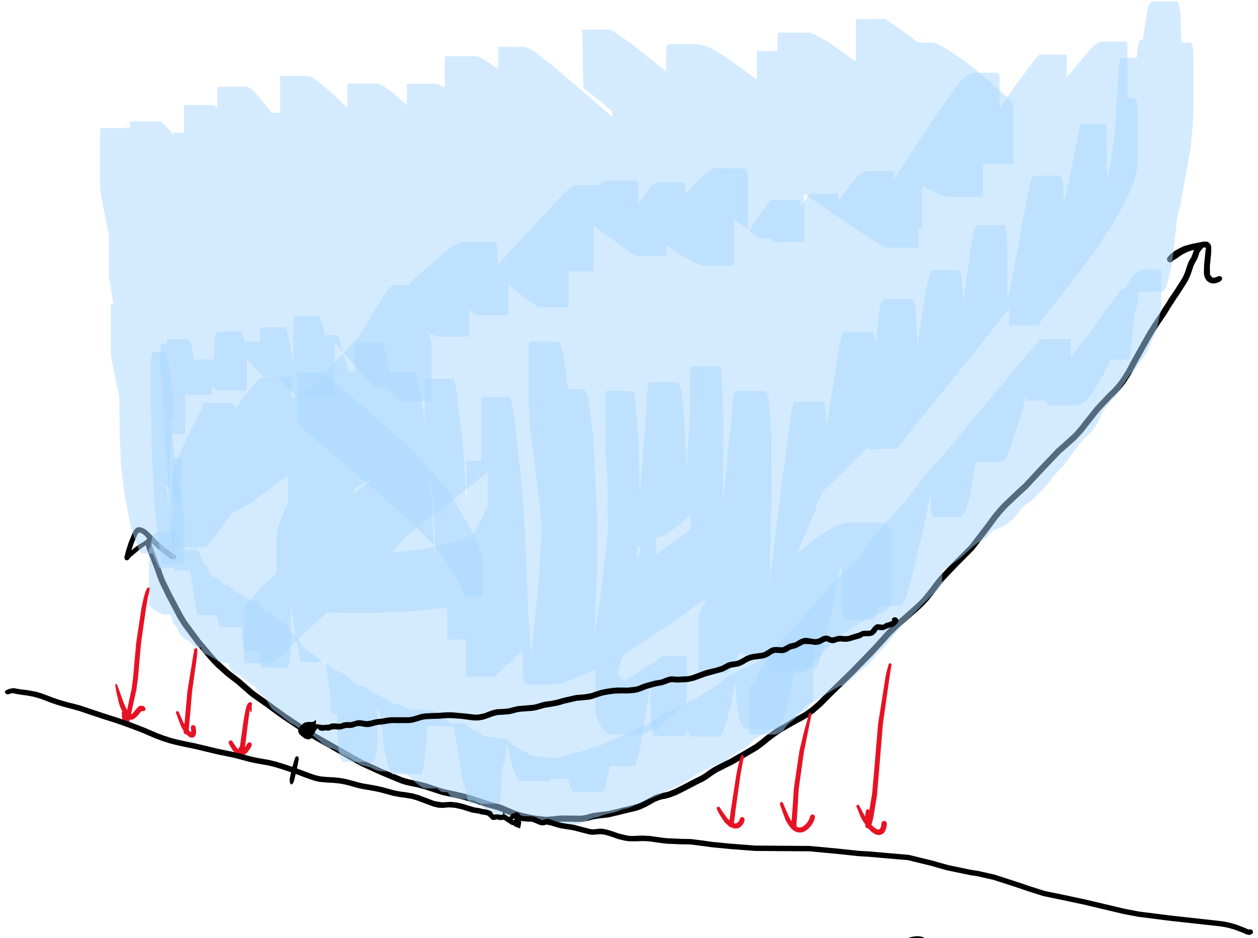
$$\forall x, y \in S$$
$$\forall \lambda \in [0,1]$$
$$\lambda x + (1 - \lambda) y \in S$$





Convex Functions

- What makes a function convex? “It curves up.”
  - Tangent line always beneath the function.
  - Derivative is increasing (second derivative positive)
  - Epigraph (set of all points above the function) is a convex set
- Affine (including linear) functions are convex and concave
- All of this generalizes to multiple dimensions



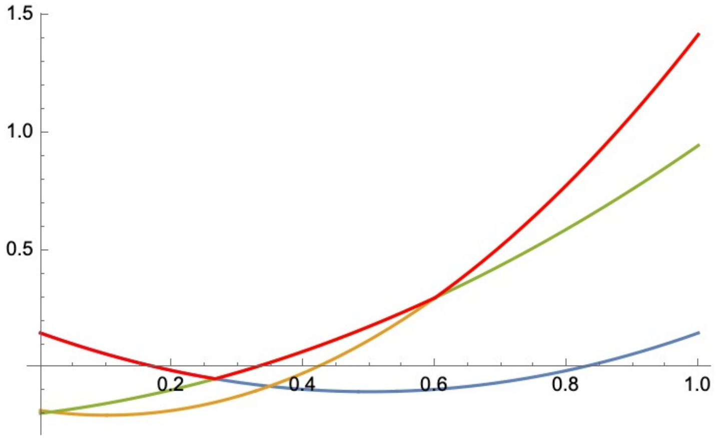
$$\frac{\partial^2 f}{\partial x^2} \geq 0$$

$\forall x_1, x_2$

$\forall \lambda \in [0,1]$

$$\lambda f(x_1) + (1-\lambda) f(x_2) \leq f(\lambda x_1 + (1-\lambda) x_2)$$

Convexity-Preserving Operations

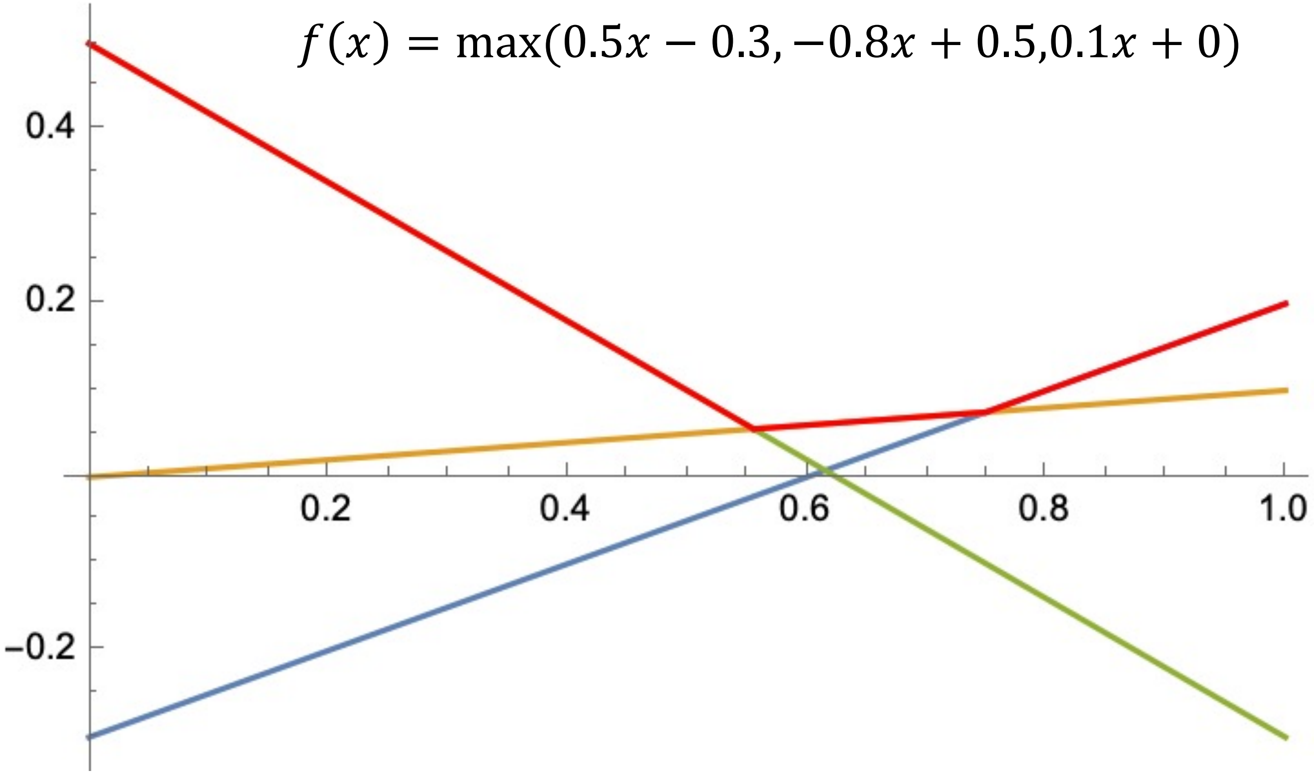


Pointwise max of 3 convex functions (red) is also convex.

- Let $f_1, f_2, \cdots, f_n$ be convex functions. Many operations preserve convexity. Key examples:
  - $f_i + f_j$ is convex
  - cf_i for positive c is convex
  - $f_i(A\,x + b)$ is convex (linear transformation)
- Crucially, the "pointwise maximum" operation is convex:
  - $g(x) = \max_i f_i(x)$ is convex
  - This will be very important, as we'll see

Pointwise Maximum of Affine Functions

$$f(x) = \max(0.5x - 0.3, -0.8x + 0.5, 0.1x + 0)$$



- Affine functions are convex, so their pointwise maximum makes a convex function (drawn in red).
- In fact, as we will see, this is an "if and only if" relationship – *every* convex function can be represented as a max over affine functions

Convex Conjugate

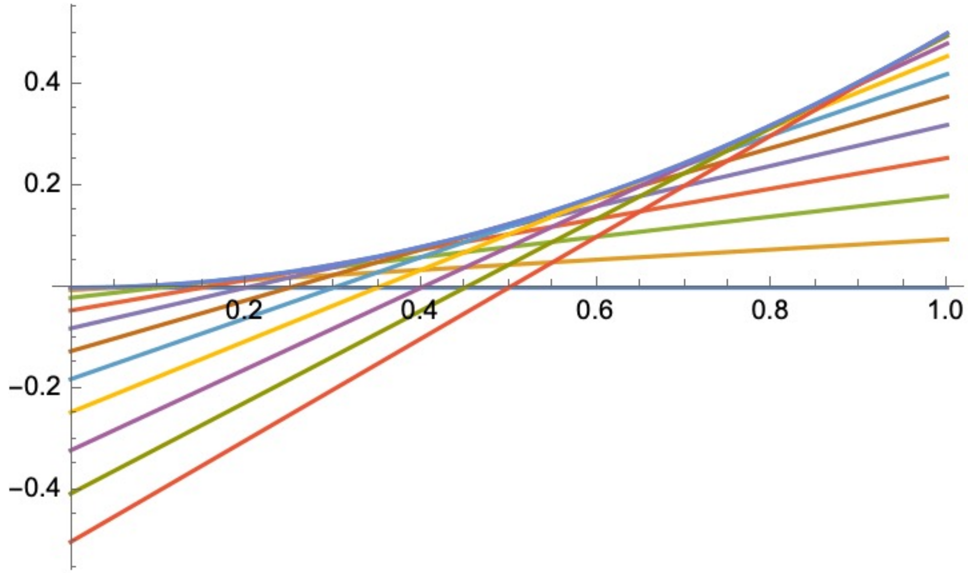
- The convex conjugate is an extremely important operation.
- It is the following operation:

$$f^*(a) = \sup_x x \cdot a - f(x)$$

- Takes a function on column vectors, produces a function on row vectors (primal/dual)
- Convex conjugate of ANY function well defined, and results in a convex function.
- We'll focus on conjugate of a convex function. In that case, f** = f.

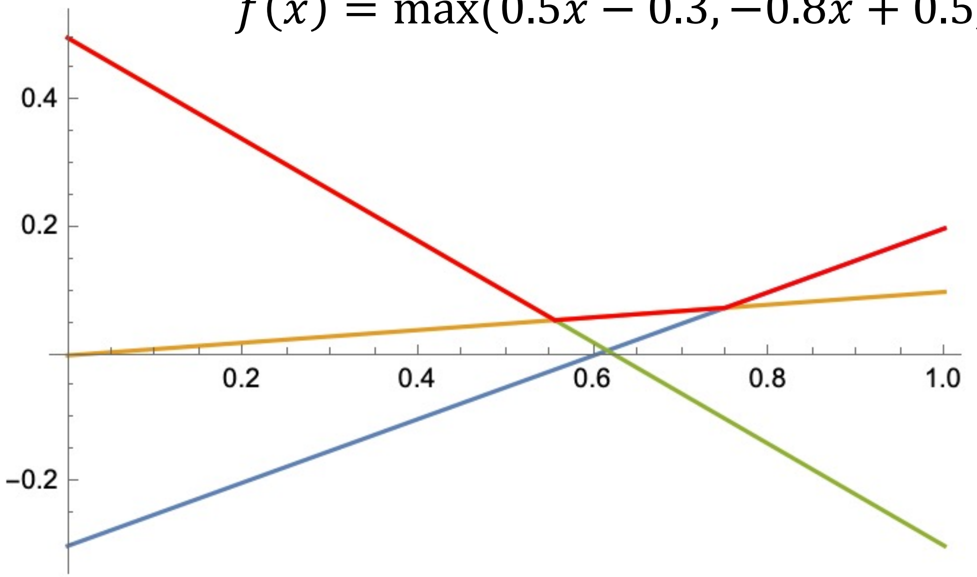
Tangent Lines and Convex Functions

- Remember one definition of convexity – tangent lines always below function
- Hand-waving argument: *all* the tangent lines, considered together, completely define your function
- Convex conjugate f* defines concrete relationship between tangent lines and f



Geometric Intuition for the Convex Conjugate

$$f(x) = \max(0.5x - 0.3, -0.8x + 0.5, 0.1x + 0)$$



Given any possible slope of a tangent line, the conjugate is going to tell you which y-intercept to assign, to rebuild f.

$$f^*(0.5) = 0.3$$

$$f^*(-0.8) = -0.5$$

$$f^*(0.1) = 0$$

Many slopes aren't found, e.g. $f^*(1) = \infty$

Non-differentiable points have many other possible tangent lines (subgradients) – let's just not worry too much about that.

Max-over-affine representation is universal

- It is a fact that for any convex function f,$f^{**} = f$.

$$f^*(a) = \sup_x x \cdot a - f(x)$$

$$f^{**}(x) = \sup_a x \cdot a - f^*(a) = f(x)$$

- Consider all points a such that $f^*(a)$ is finite. Each of those defines an expression $x \cdot a - f^*(a)$ which is linear as a function of x
- $f(x)$ can be written as a max over all those linear expressions – and this holds for every convex function.

Back to mechanisms

- Write down our truthful direct revelation mechanism as a convex utility function u(x).
  - Allocation a(x) = u'(x) (slope), payment u'(x)*x – u(x) (y-intercept)
- The convex conjugate then defines a function $u^*(a)$ mapping outcomes to payments – the menu.
- Taking the conjugate again recovers the original utility function.

Concrete Example

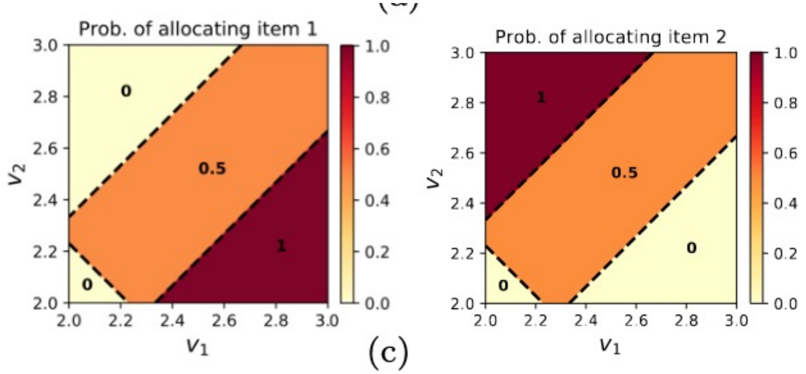
- $u(x) = \max\big((1,0) * x - 0.5, (0,1) * x - 0.6, (1,1) * x - 0.95, 0\big)$
- $u^*\big((1,0)\big) = 0.5$
- $u^*\big((0,1)\big) = 0.6$
- $u^*\big((1,1)\big) = 0.95$

Conclusion

- This always works, for any convex utility function.
- So: direct revelation mechanisms have convex utility functions, which can be turned into menus
- Menus have corresponding convex utility functions
- Thus, we’ve completely characterized feasible mechanisms and connected them to feasible menus

Optimal mechanism design

- Next lecture – choosing optimal mechanisms
- Mixed bundling is allowed, but is it necessary?
- Menu-size complexity?
- Checking optimality?
- Machine learning for finding optimal mechanisms



(c)