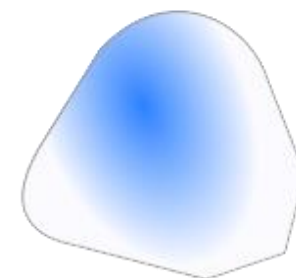
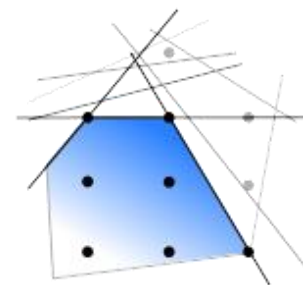
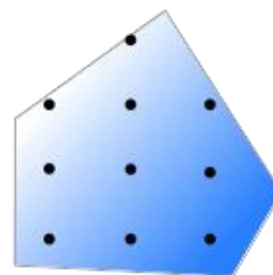
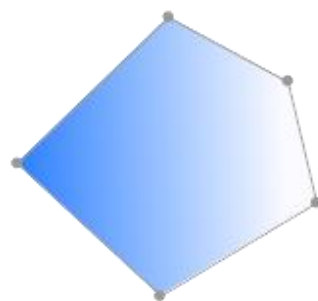
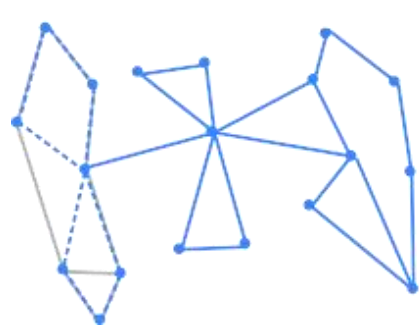
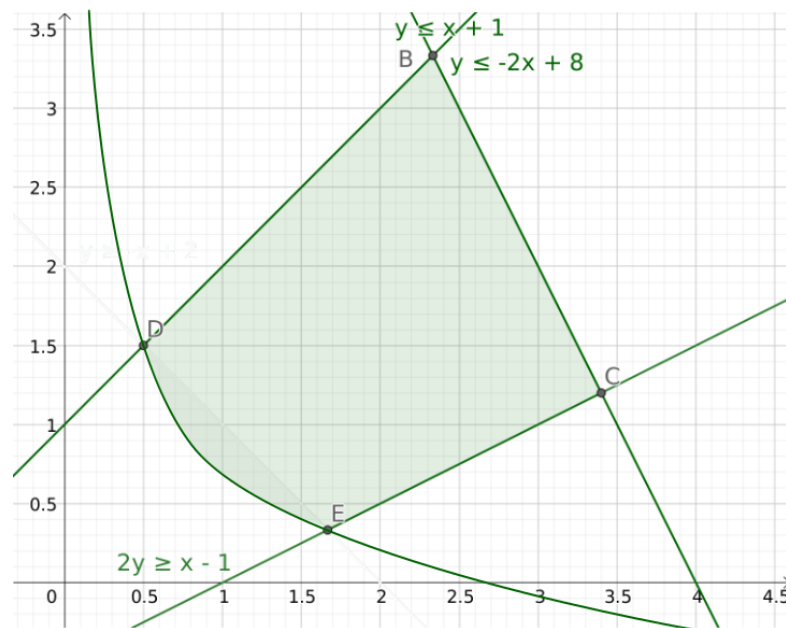


# Convex Optimisation 2

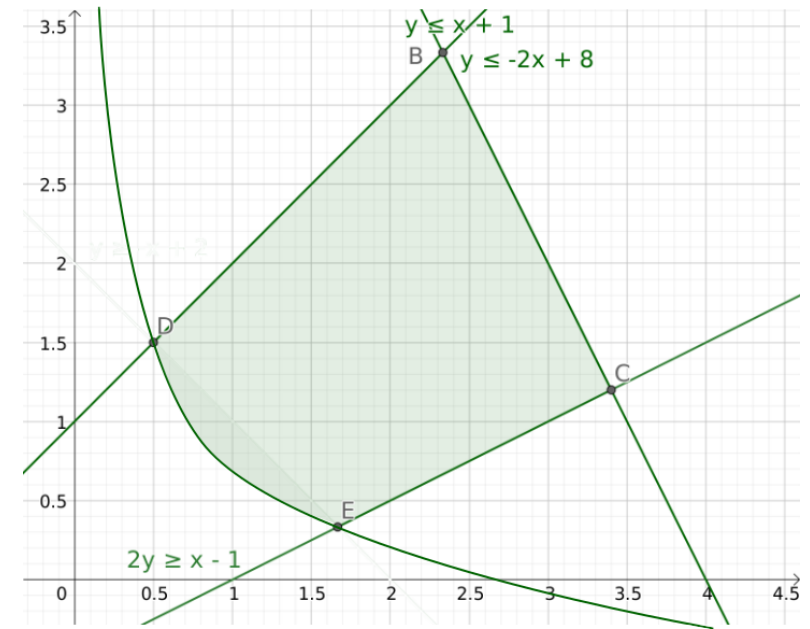
COMP4691 / 8691



# Convex Optimisation Outline

- Convexity
- Unconstrained Optimisation
  - Optimality
  - Gradient Descent
  - Newton's Method
  - Gradient Projection
  - Penalty Method
- Constrained Optimisation
- Lagrangian Duality
- KKT Conditions
- Interior Point Method

In some cases we will look at more general non-convex problems, as some of the theory applies there also.



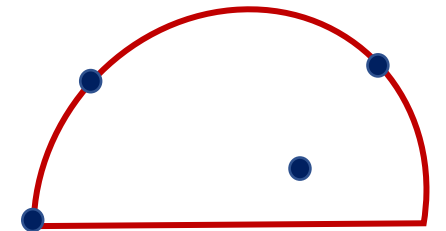
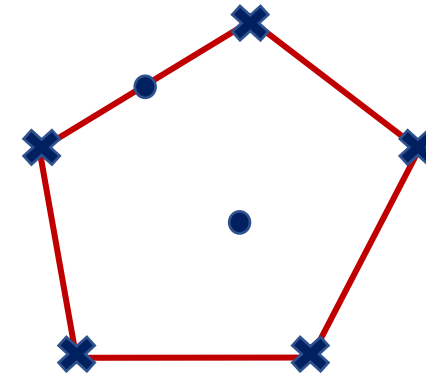
# Convex Optimisation

The objective is a *convex function* (for min) over a convex feasible set.

Linear programming fits these requirements.

In general **compared to LP**:

- Feasible region need not be a polyhedron
- Optimal solution can be anywhere on the boundary of the feasible region (and need not be at an extreme point)
- Optimal solution can be in the interior of the feasible region



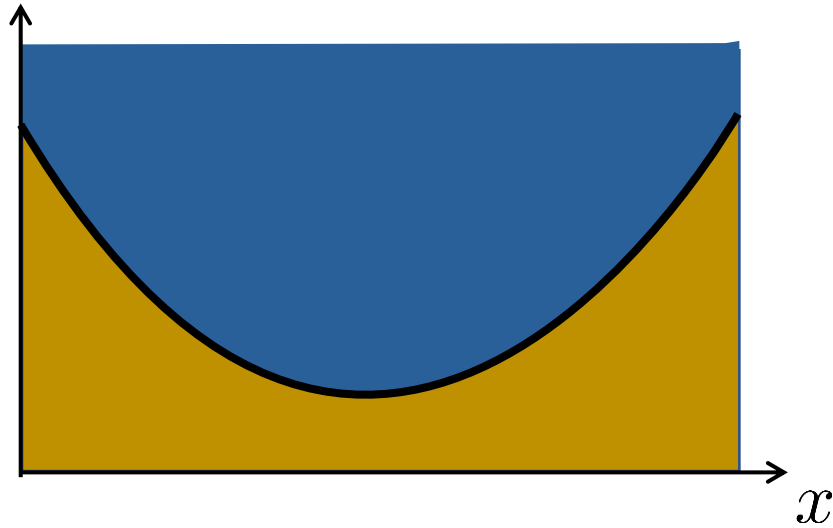
Simplex algorithm won't work.

# Convex Functions

A **convex** function's *epigraph* is a convex set.

A **concave** function's *hypograph* is a convex set.

**Epigraph** (**hypograph**) = set of points on or **above** (**below**) the *graph* of a function.



Note that might only be interested in the convexity of a function over an interval or convex set (single or multi-variate) rather than the full space.

# Vector Calculus: Gradient

The gradient is a generalisation of the derivative to multivariate functions

For a differentiable scalar function:  $f : \mathbb{R}^n \longrightarrow \mathbb{R}$  A vector that gives the direction of greatest increase in the function and strength of that increase.

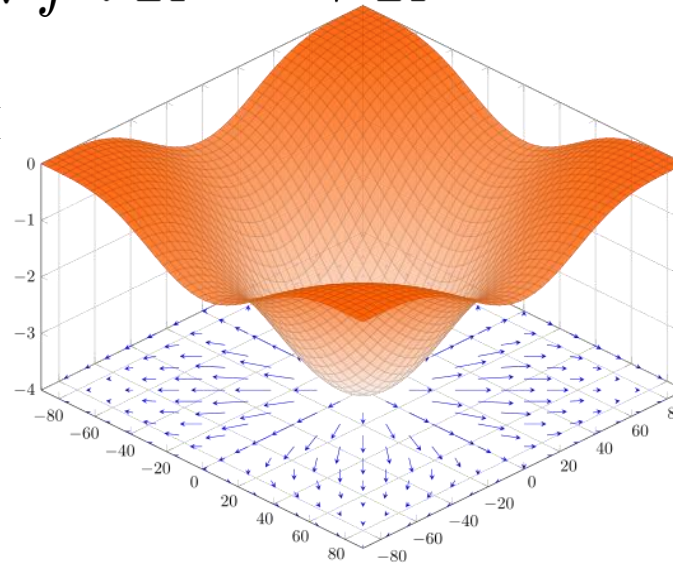
$\nabla$  "nabla" / "del"

$\nabla f$  the **gradient** of  $f$ :  $\nabla f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$\nabla f(x)$  the gradient of  $f$  at  $x$

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$$

$\nabla f(x)^\top u$  the rate of change of the function in the direction of the unit vector  $u$



In the hills analogy, it gives the direction that would lead us up the slope and the steepness of that slope.

$$\nabla f(x) = 0?$$

# Convexity using Hessian

The Hessian is the Jacobian of the gradient:  $H_f = J_{\nabla f}$

## Multi-variate function

Convex iff: Hessian matrix is **positive semidefinite** everywhere

$$A \succeq 0 \quad \text{iff} \quad x^\top A x \geq 0 \quad \forall x \in \mathbb{R}^n$$

Strictly convex iff: Hessian matrix is **positive definite** everywhere

$$A \succ 0 \quad \text{iff} \quad x^\top A x > 0 \quad \forall x \in \mathbb{R}^n \setminus 0$$

$$f(x, y) = x^2 + y^2 + 4xy \quad \text{Is } f \text{ convex?}$$

$$H_f = \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}$$

**Constant, so only one Hessian to evaluate**

# Unconstrained Optimality

An **unconstrained, twice-differentiable** function has a local minimum at a point if:

$$\boxed{\nabla f(x^*) = 0} \quad H_f(x^*) \succeq 0$$

**Together** these are **sufficient** conditions for a local minimum.

**Separately they are only necessary**, but not sufficient.

First-order **necessary** condition for unconstrained problem local optimum

Since a convex function has a positive semidefinite Hessian everywhere:

If the function is convex, then first-order condition becomes a **sufficient** condition

# Gradient Descent

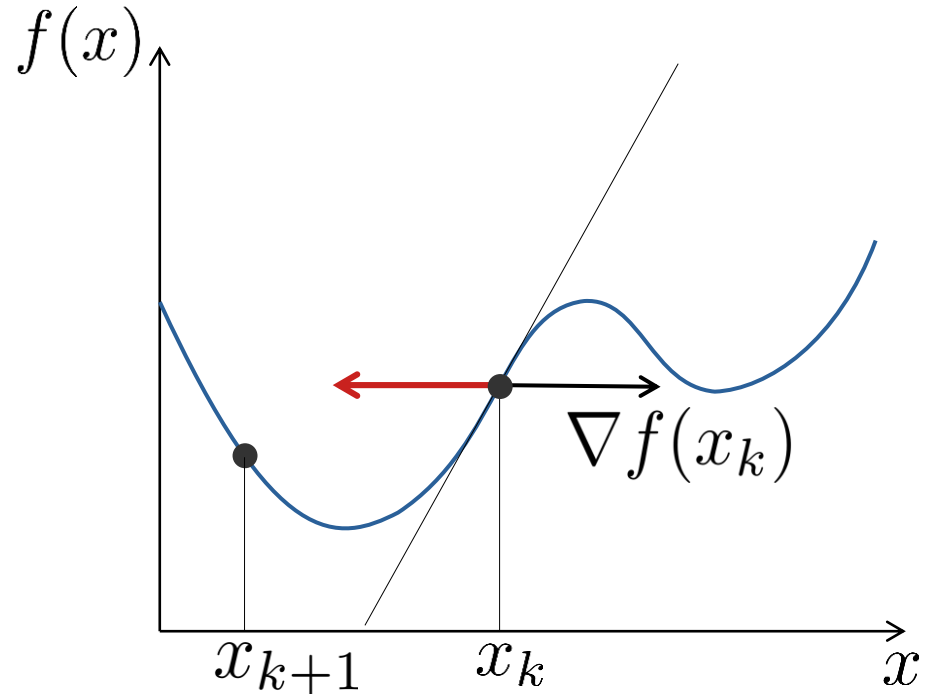
For unconstrained differentiable problems.

$$\min_{x \in \mathbb{R}^n} f(x) \quad f(x) \text{ differentiable}$$

$x_{k=0}$  a starting value

$\nabla f(x_k)$  the gradient for the k-th iterate

Points in the direction that would take us up the steepest part of the hill, and has magnitude equal to that slope.



If we are minimising, we need to step in the opposite direction:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \quad \text{for some step size } \alpha_k$$



# Methods

Gradient descent is a **first-order** method (recall general statements in rate of convergence slides).

**First-order** methods locally approximate function as being linear, in deciding the step direction (using **first derivative** information).

**Second-order** methods locally approximate function as being quadratic, in deciding the step direction (using **second derivative** information).

# Newton's (Raphson) Method

Newton's method is commonly used for **finding the zeros** of non-linear functions. Start with Taylor series of a **multivariate scalar function**:

$$f(x) = f(a) + (x - a)^\top \nabla f(a) + \frac{1}{2}(x - a)^\top H_f(a)(x - a) + \dots$$

If we evaluate about our current iterate and set  $x$  to be our next:

$$a = x_k \qquad x = x_{k+1}$$

Truncating the Taylor series we get:

$$f(x_{k+1}) \approx f(x_k) + (x_{k+1} - x_k)^\top \nabla f(x_k)$$

We want to move to the point where:

$$f(x_{k+1}) = 0 \qquad \implies 0 \approx f(x_k) + (x_{k+1} - x_k)^\top \nabla f(x_k)$$

Newton's method for **finding zeros** is then:

$$\nabla f(x_k)^\top (x_{k+1} - x_k) = -f(x_k)$$

# Newton's Method

For finding roots of  $f$ :

$$\nabla f(x_k)^\top (x_{k+1} - x_k) = -f(x_k) \quad \text{if } f \text{ scalar valued}$$

$$\boxed{J_f}(x_k)(x_{k+1} - x_k) = -\boxed{f}(x_k) \quad \text{if } f \text{ vector valued}$$

But we want to find **stationary points not zeros!** **Ideas?**

In order to do this we can replace  $f$  with its gradient (assuming it is twice differentiable):

$$J_{\nabla f}(x_k)(x_{k+1} - x_k) = -\nabla f(x_k) \quad \text{Remember Jacobian of}$$

$$\implies H_f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k) \quad \text{gradient is Hessian.}$$

When adapted to finding zeros of the gradient, it becomes a **second-order** method.

# Newton's Method

In order to have a **unique** solution for the Newton step, we need the Hessian to be invertible:

$$x_{k+1} - x_k = -H_f(x_k)^{-1} \nabla f(x_k)$$

**Positive definite** matrices are always **invertible**. The implications of this are that a **strictly** convex function will always have a well defined Newton step. If the Hessian is positive **semidefinite**, (function convex but not strictly), then we can have an infinite number of solutions.

**Does this make sense?**  $x^\top H_f(x_k) x \geq 0 \quad \forall x \in \mathbb{R}^n$  (pos semidefinite)

This allows our function to have linear segments, where a second derivative doesn't contain any information, so a second-order method doesn't make much sense! For a single-variate function:

$$x_{k+1} - x_k = -\frac{f'(x_k)}{f''(x_k)}$$

Trouble when second derivative is zero (locally linear). Best we can do is a gradient step.

# Newton's Method

$$x_{k+1} - x_k = \boxed{-H_f(x_k)^{-1} \nabla f(x_k)} \longleftarrow \alpha_k d_k$$

It provides both a direction and a step size. We may want to adjust that step size to improve convergence: we are locally **approximating** the function as being **quadratic** to get our direction and step size but in reality it might not be! Again we can use techniques like **line search**.

What if it **is** quadratic?  $f(x) = \frac{1}{2}x^\top Qx + c^\top x$   $Q$  pos definite

$$\nabla f(x) = Qx_k + c \quad H_f(x_k) = Q$$

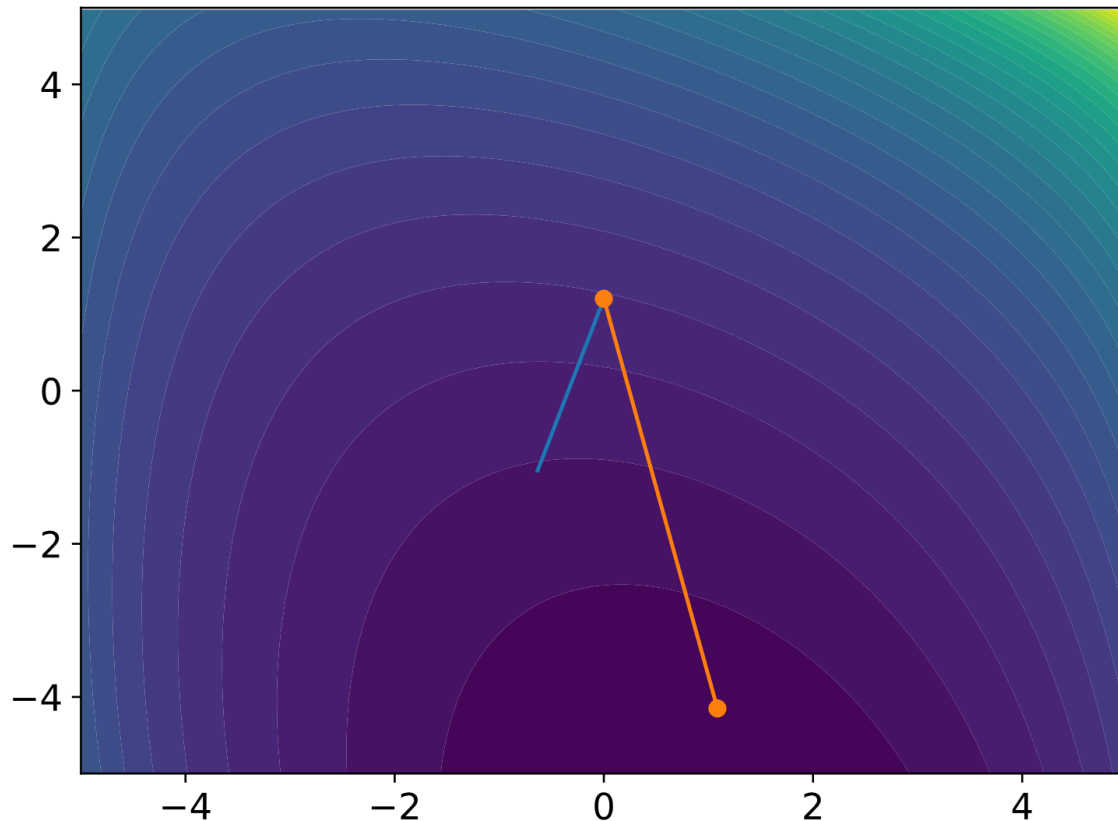
$$H_f^{-1}(x_k) \nabla f(x_k) = Q^{-1}Qx_k + Q^{-1}c = x_k + Q^{-1}c$$

$$(x_{k+1} - x_k) = -x_k - Q^{-1}c \quad \implies x_{k+1} = -Q^{-1}c$$

We go straight to the minimum in one step.

# vs Gradient Direction

$$f(x, y) = -\log(x + 6) - \log(-x + 6) - 2\log(-x - 2y + 16) - 1.2\log(1.2x + y + 16))$$



Gradient Step  $\alpha = 10$

Newton Step

# Newton's Method

Second-order method, so typically superlinear convergence.

If problem is **not** convex:

We might converge to a local **min**, **max** or **saddle point**, by default we can't choose which. For gradient descent the second two would be unstable. We can use the definiteness of the Hessian to see which we have obtained.

# Non-differentiable?

There are dedicated methods that can work with **subgradients** if the problem is non-differential.

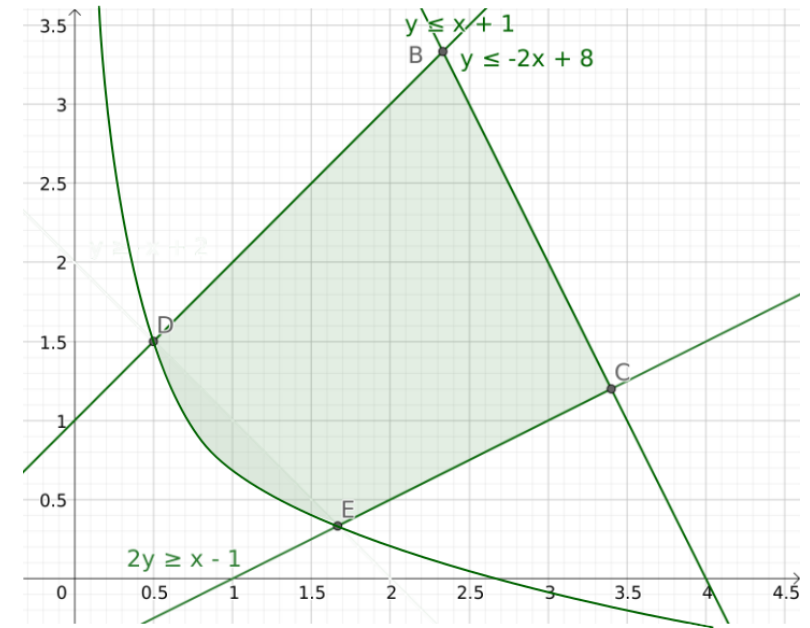
Often the non-differentiable parts can be dealt with by a change in formulation when we allow constraints:

$$\min_x |x| \quad \longrightarrow \quad \begin{array}{ll} \min_{x,y} & y \\ \text{s.t.} & y \geq x \\ & y \geq -x \end{array}$$



# Convex Optimisation Outline

- Convexity
- Unconstrained Optimisation
- Constrained Optimisation
  - Gradient Projection
  - Penalty Method
- Lagrangian Duality
- KKT Conditions
- Interior Point Method

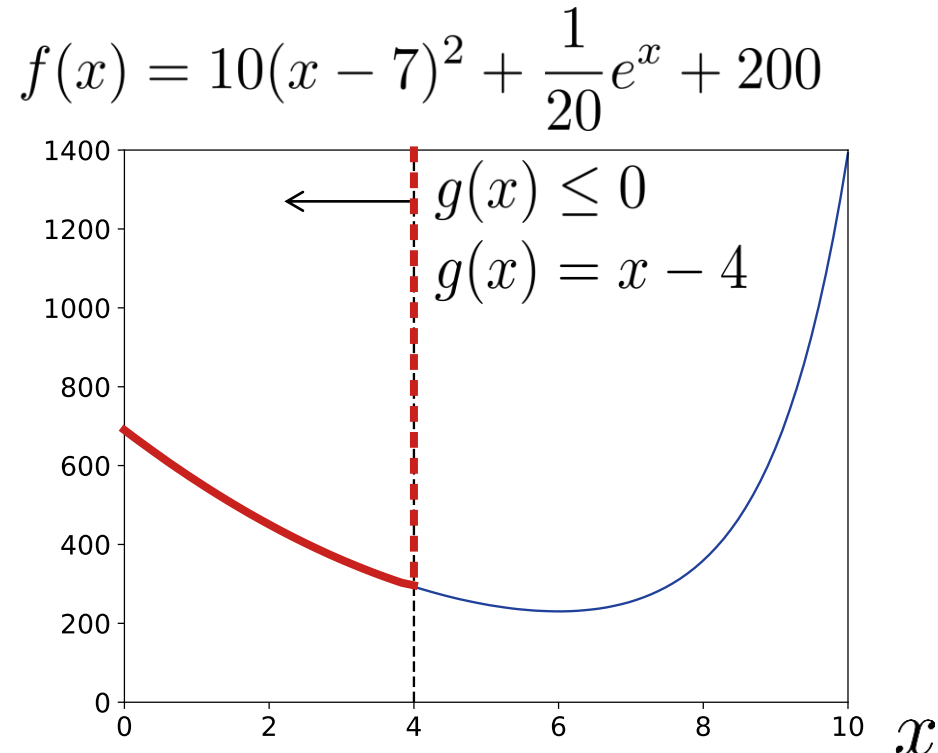


In some cases we will look at more general non-convex problems, as some of the theory applies there also.

# Constrained Optimisation

So far we have assumed it is possible to pick any  $x$ . Using the above techniques we run into trouble:

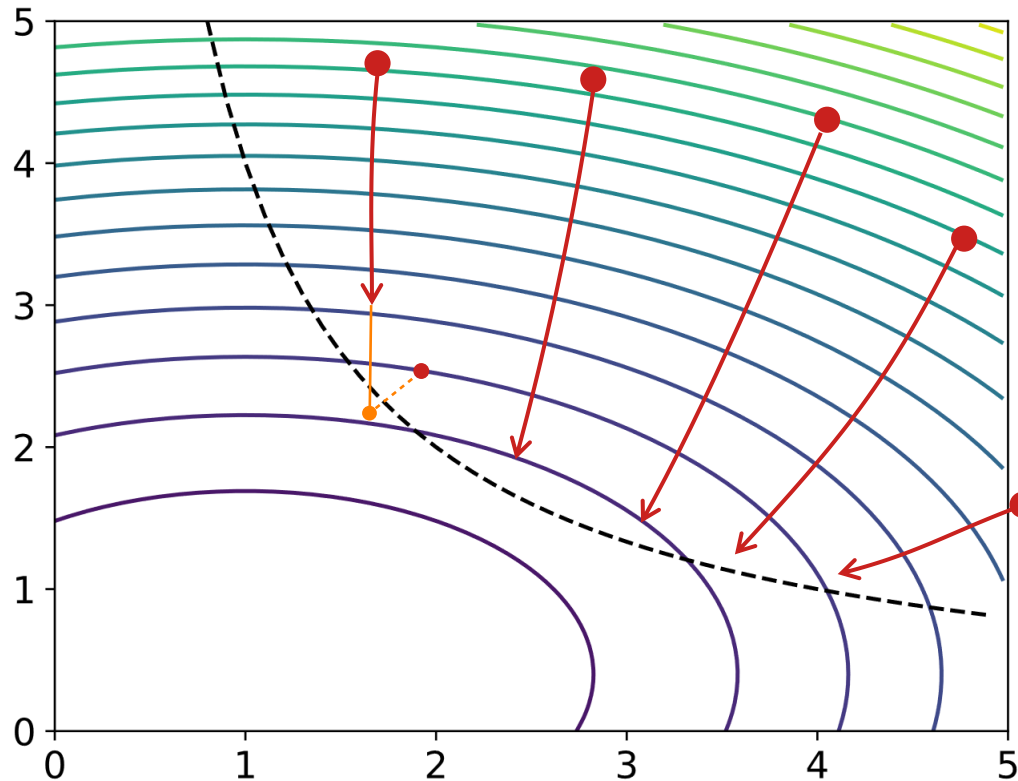
Constraints create hard boundaries (effectively a step change with infinite cost), that by default don't give us any warning of their presence until we hit them.



Can we just make gradient steps until we hit a constraint, i.e. stop at the point we first touch boundary? Will that give us an optimal solution?

**Here:** probably yes  
**In general:** no

# Following Gradient to Boundary



Consider various starting points

We would need some way to move along the boundary

Stopping at the intersection of the gradient step with the boundary doesn't work, but if we can take a **projection** of our gradient step back onto our **feasible set** we can make progress.

Such techniques are **gradient projection methods**. It can require extra effort to compute (exact) projections and convergence can be slow.

# Constrained Optimisation

Many different techniques have been developed to solve constrained convex optimisation problems.

Particular algorithms may only be suitable when we place further restrictions on our functions and / or derivatives, e.g., among:

Lipschitz continuity

Smoothness

Convexity

Strict Convexity

Strong Convexity

These can also impact the convergence rates for a particular algorithm.

Let's look at one more simple (but rarely used in practice) approach, before getting stuck into the theory for the more advanced techniques.

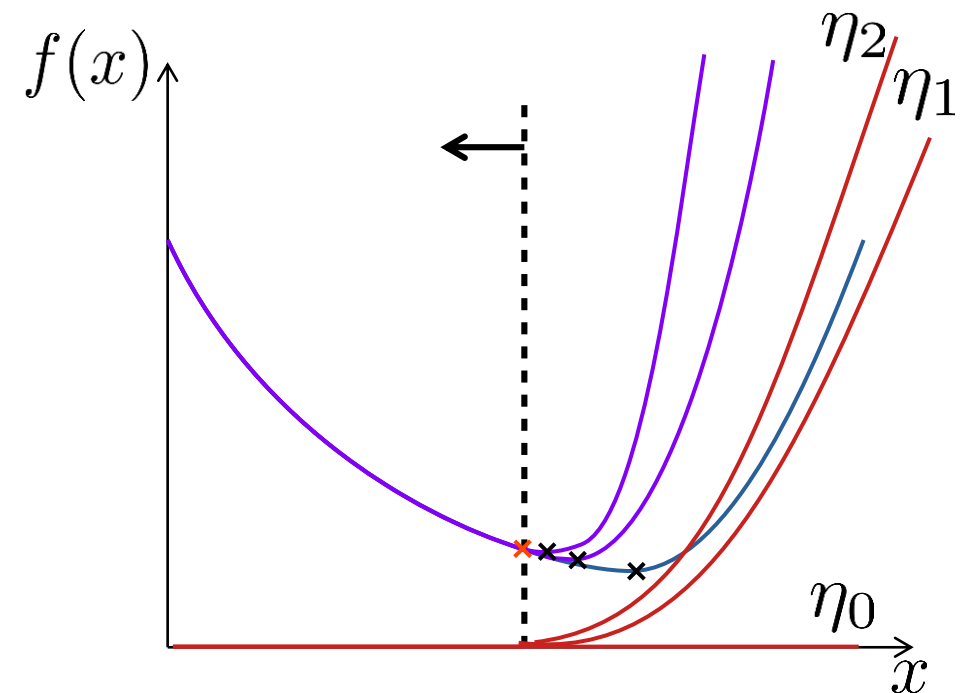
# Penalty Method

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \quad x^{(k)} = \arg \min_x f(x) + \textcolor{red}{\eta_k} \|\max(0, g(x))\|_2^2 + \textcolor{red}{\eta_k} \|h(x)\|_2^2$$

Don't try to stay in feasible region, instead penalise violation and gradually bring iterates back to feasibility.

Solve unconstrained problem for some **penalty**. If the constraints are violated, increase the penalty and solve again (using the iterate as starting point).

If the penalty is initially too severe, we will potentially jump all over the place using something like gradient descent. The penalty needs to be gradually increased.



# Penalty Method

$$f(x) = 10(x - 7)^2 + \frac{1}{20}e^x + 200$$

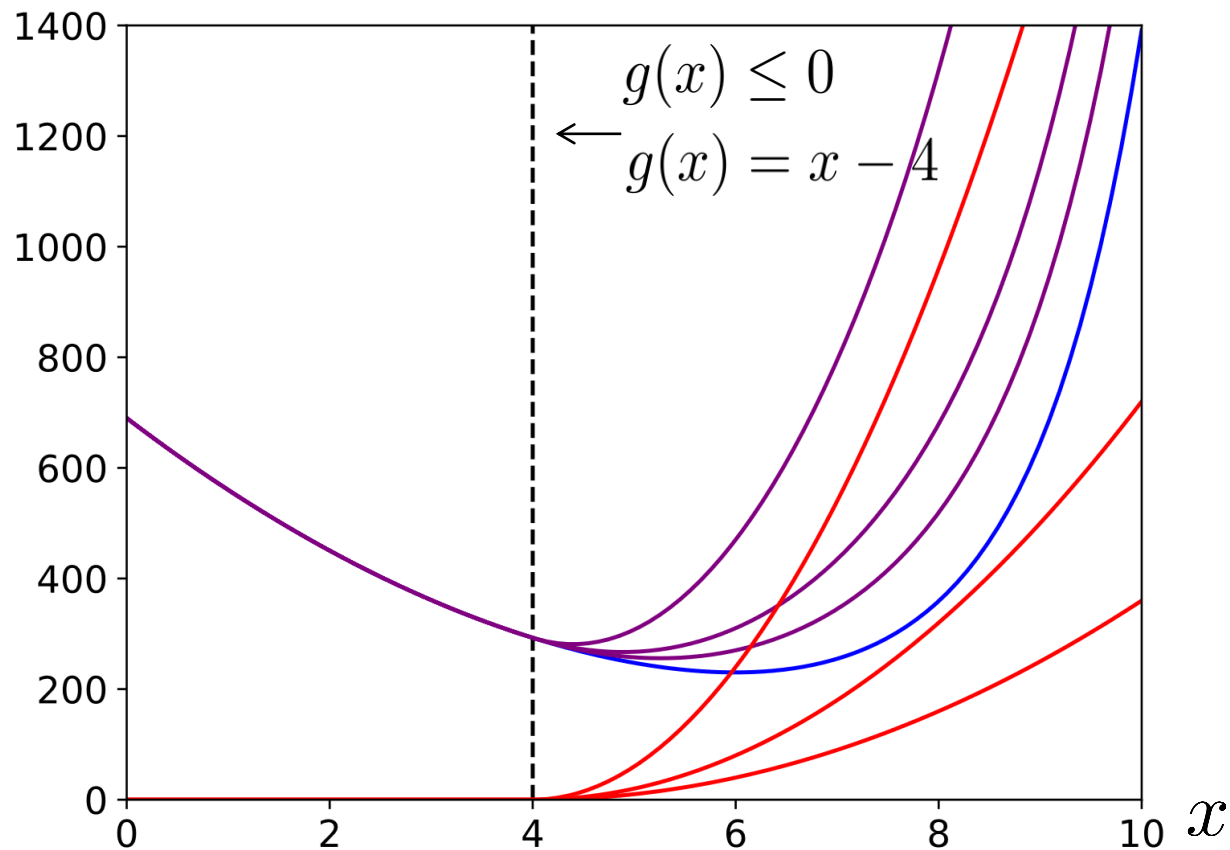
$$\eta = 60$$

The penalty needs to approach infinity in order to satisfy the constraint

$$\eta = 20$$

$$\eta = 10$$

Details omitted... we can do something better...



# Constrained Optimisation

We can use a more principled approach to incorporating constraints.

For unconstrained optimisation we relied on the necessary condition that a local minimum will be at a point with zero gradient. This is no longer necessary!

We'd like to derive some analogous necessary conditions for optimality in a constrained context.

First we need to understand **Lagrangian Duality** in more detail, so let's first take a detour of that.

# Lagrangian Relaxation

Just like with duality in LPs, we are going to start off **generating lower bounds** to our optimisation problem. Assume we have a single constraint ( $g$  is a scalar valued function):

$$\min_x f(x) \qquad \min_x f(x) + \mu g(x) \qquad \text{for some } \mu \geq 0$$

$$\text{s.t. } g(x) \leq 0$$

This is a **Lagrangian relaxation** of our constraint.

$$\min_x f(x) \quad \begin{matrix} \leq \\ = \\ \geq \end{matrix} \quad \min_x f(x) + \mu g(x) \quad \begin{matrix} \text{Assuming the problem is feasible with} \\ \text{optimal solution: } x^* \end{matrix}$$

$$\text{s.t. } g(x) \leq 0$$



?

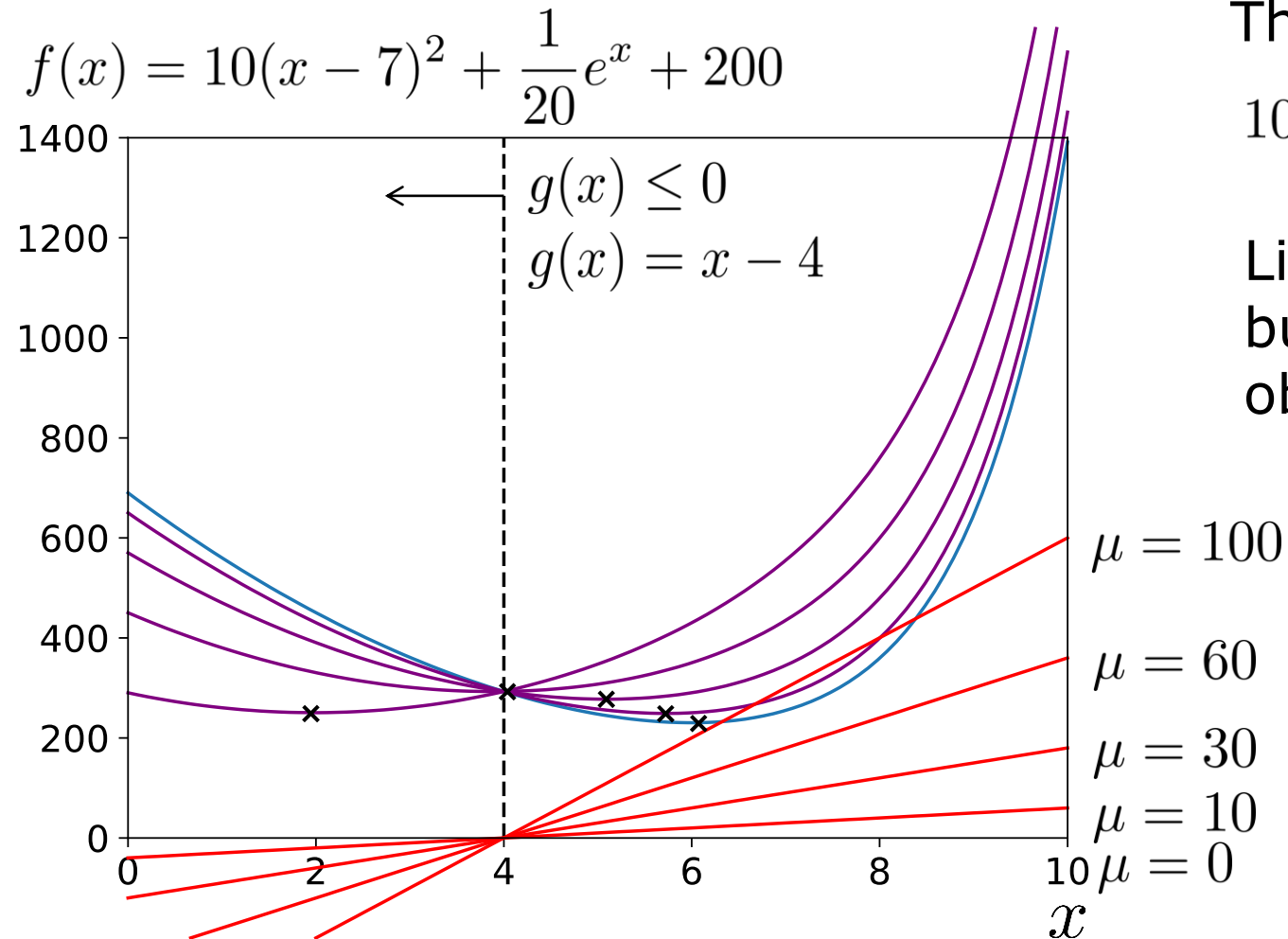
The relaxation can always pick this point which will be at least as good:  $f(x^*) \geq f(x^*) + \mu g(x^*)$

Because

$$\begin{aligned} g(x^*) &\leq 0 \\ \mu &\geq 0 \end{aligned}$$



# Lagrangian Relaxation



The Lagrangian relaxation of  $g$ :

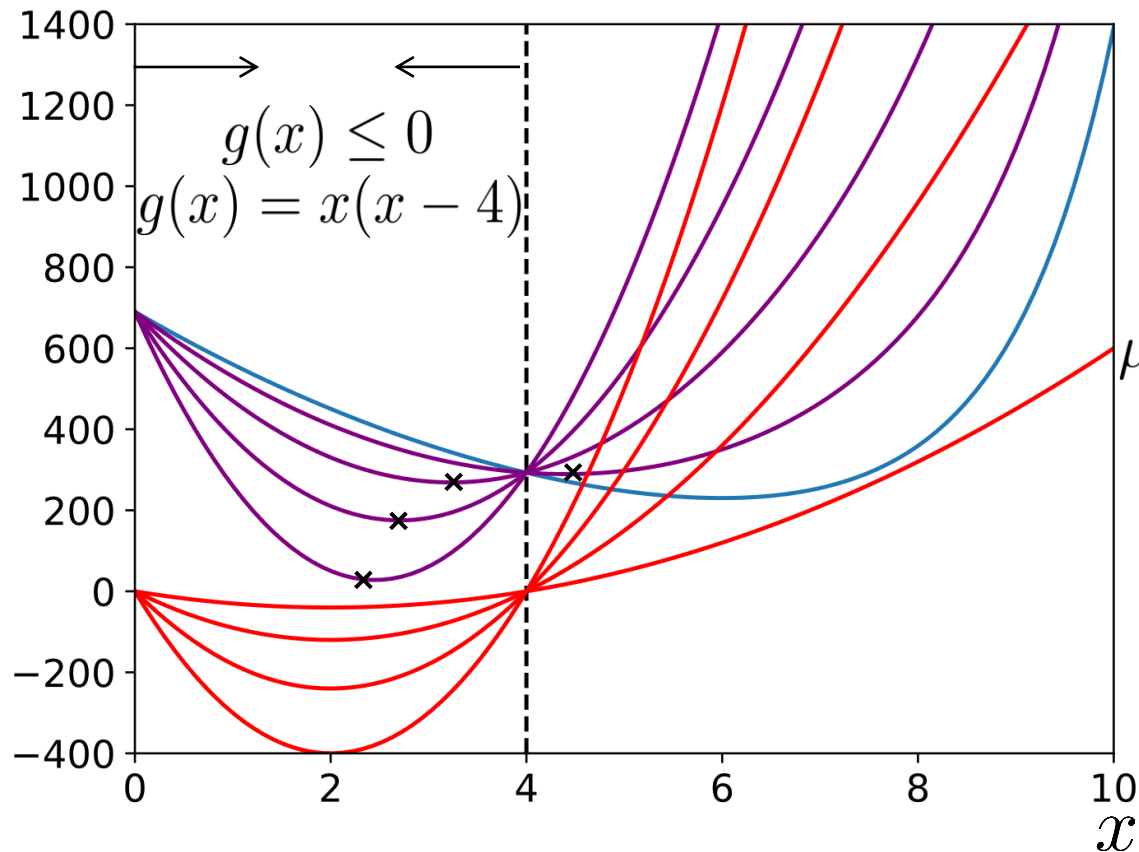
$$10(x - 7)^2 + \frac{1}{20}e^x + 200 + \mu(x - 4)$$

Like the penalty method penalty, but linear, and also impacts objective within feasible region.

This hints at an alternative algorithm for solving constrained problems.

# Lagrangian Relaxation

$$f(x) = 10(x - 7)^2 + \frac{1}{20}e^x + 200 \quad \mu = 100 \quad \mu = 60 \quad \mu = 30$$



Where constraint is nonlinear:  
 $10(x - 7)^2 + \frac{1}{20}e^x + 200 + \mu x(x - 4)$

# Lagrangian Dual Problem

$$\begin{array}{ll}\min_x & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0\end{array}$$

**Dual variables**, one for each constraint again

Also known as **Lagrange** or **KKT multipliers**

Lagrangian Function:

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^\top g(x) + \lambda^\top h(x) \quad \mu \geq 0$$

Lagrangian Dual Function:

$$d(\mu, \lambda) = \inf_x \mathcal{L}(x, \mu, \lambda) \quad \text{The dual function is **always** concave!}$$

Dual Problem: like a min, see next slide

$$\max_{\mu \geq 0, \lambda} d(\mu, \lambda) = \max_{\mu \geq 0, \lambda} \inf_x \mathcal{L}(x, \mu, \lambda) \quad \text{Find tightest lower bound}$$

# Why Infimum / Supremum?

**Infimum** of a set: largest lower bound on values in set

**Supremum** of a set: smallest upper bound on values in set

Similar to minimum and maximum, but value doesn't have to be part of the set:  
 $\inf\{x \mid x > 5\} = 5$

$$\begin{aligned} \min_x e^x \\ \text{s.t. } x \geq -5 \end{aligned}$$

$$\mathcal{L}(x, \mu) = e^x - \mu(5 + x) \quad \mu \geq 0$$

$$d(\mu) = \inf_x \mathcal{L}(x, \mu)$$

$$d(0) = \inf_x e^x = 0$$

This problem has a well defined minimum of:  $e^{-5}$

Exponential has no minimum, need infimum

We have them for precision, but for the most part you can just read them as min and max.

# Lagrangian Dual Problem

**Weak duality** holds:  $\max_{\mu \geq 0, \lambda} d(\mu, \lambda) \leq \min_x f(x)$   
s.t.  $g(x) \leq 0$

Functions don't need to be differentiable, convex, or even continuous.  $h(x) = 0$

In order to get **strong duality** we need some more conditions.

One example is that strong duality holds for a convex optimisation problem when **Slater's Condition** holds.

For convex problems, Slater's Condition is **sufficient**, but **not necessary** for **strong duality** to hold (there are other conditions).

For **most convex** optimisation problems you come across will have **strong duality**.

# Slater's Condition (1950)

$$\min_x f(x)$$

$$\text{s.t. } g(x) \leq 0$$

$$h(x) = 0 \quad \longleftarrow \text{generally should be affine for convex problem}$$

For convex optimisation problems it requires that the:

Interior of feasible region is not empty... considering the constraints that are **not** affine / linear:

Assuming all the  $g$  constraints are nonlinear, then there exists at least one point  $x^*$  where:

$$g(x^*) < 0$$

$$h(x^*) = 0$$

# Concavity of Dual Function

For convex / concave functions we have:

$f(x, y)$  concave in  $x$ , then  $g(x) = \inf_{y \in Y} f(x, y)$  is concave in  $x$

$f(x, y)$  convex in  $x$ , then  $g(x) = \sup_{y \in Y} f(x, y)$  is convex in  $x$

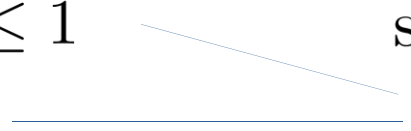
where  $Y$  can be nonconvex (**I'm not going to prove this here**).

Lagrangian is affine in the dual variables (concave and convex), therefore the first relation above tells us that the dual function is concave.

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^\top g(x) + \lambda^\top h(x) \qquad d(\mu, \lambda) = \inf_x \mathcal{L}(x, \mu, \lambda)$$

# Dual Function Example

Let's look at an example of this rather than digging into the proof.

$$\begin{array}{ll} \min_x (x - 0.6)^2 & \min_x (x - 0.6)^2 \\ \text{s.t. } 0 \leq x \leq 1 & \text{s.t. } 0 \leq x \leq 1 \\ x \in \mathbb{Z} & x(x - 1) = 0 \\ & x \in \mathbb{R} \end{array}$$


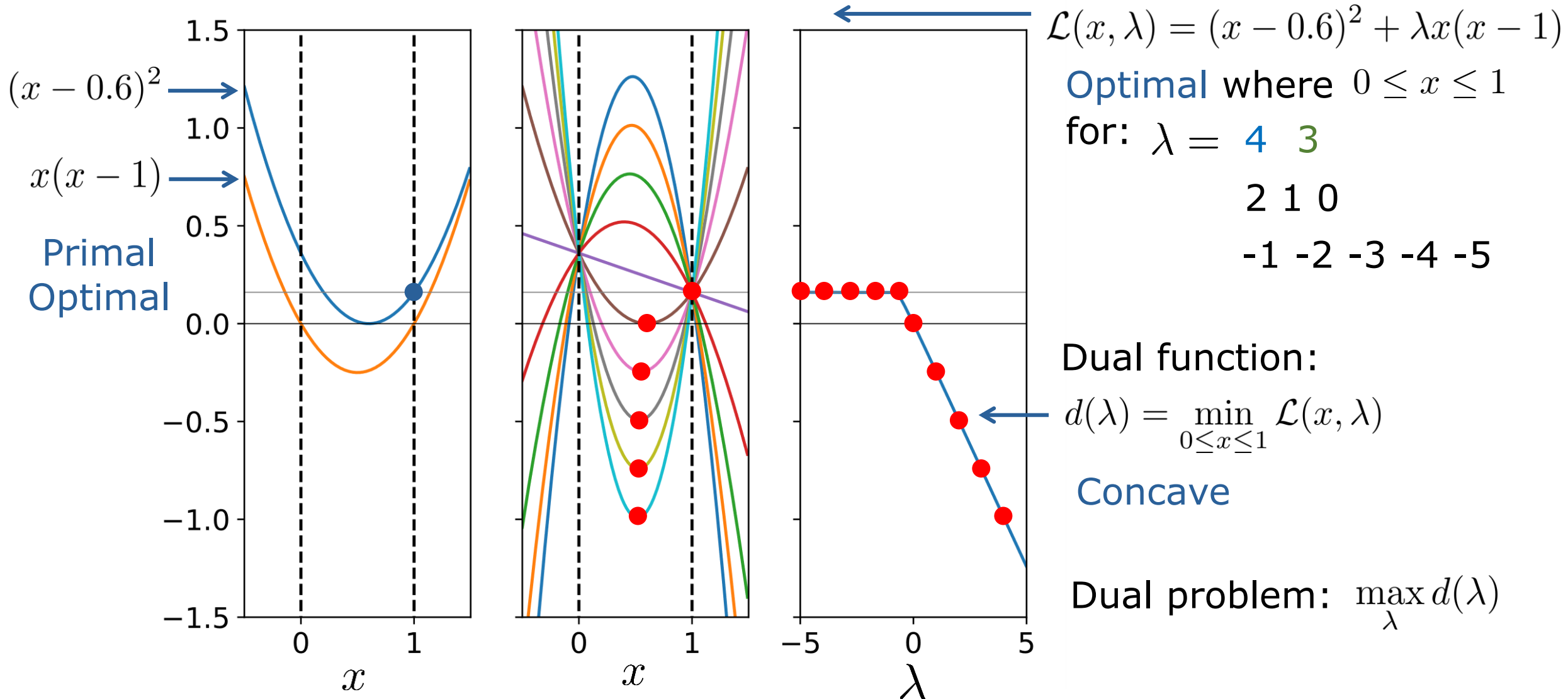
Reformulate integrality  
requirement as a nonconvex  
constraint

$$\begin{array}{ll} \max_{\lambda \in \mathbb{R}} \min_{x \in \mathbb{R}} (x - 0.6)^2 + \lambda x(x - 1) & \\ \text{s.t. } 0 \leq x \leq 1 & \end{array}$$

“Dual problem” (to keep simple we  
only relax the nonconvex  
constraint)



# Dual Function Example



# Dual Function Example

$$\begin{aligned} \min_x \quad & (x - 0.6)^2 \\ \text{s.t.} \quad & 0 \leq x \leq 1 \\ & x \in \mathbb{Z} \end{aligned}$$

Dual problem optimal is 0.16, which is equal to the primal problem optimal.

The duality gap is zero! **Strong duality** holds for this simple nonconvex problem (but in general not other nonconvex problems).

What is the **linear relaxation** optimal?

$$x = 0.6, \text{ objective} = 0$$

A Lagrangian relaxation would be tighter for any  $\lambda < 0$  (don't need to optimally solve the dual function to get an improvement over linear relax)