How to Deal with Quadratic Programs

'Cause they have a habit of popping up everywhere

Quadratic Programming

A quadratic program is an optimizatio problem in the form

$$\underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} x^T P x + q^T x \mid Ax \leq b \right\}$$

- $\boldsymbol{x}$ is a $\boldsymbol{n}$-dimensional decision variable
- $\boldsymbol{P}$ is a real symmetric matrix
- $\boldsymbol{q}$ and $\boldsymbol{b}$ are real vectors

Quadratic Programming

A quadratic program is an optimizatio problem in the form

$$\underset{x}{\operatorname{argmin}}\left\{\frac{1}{2}x^T Px + q^T x \mid Ax \leq b\right\}$$

- $\boldsymbol{x}$ is a $\boldsymbol{n}$-dimensional decision variable
- $\boldsymbol{P}$ is a real symmetric matrix
- $\boldsymbol{q}$ and $\boldsymbol{b}$ are real vectors

Our optimization form is (simpler) special case of QP, in the form:

$$\underset{x}{\operatorname{argmin}}\left\{\frac{1}{2}x^T Px + q^T x \mid x \geq 0\right\}$$

- Where $\boldsymbol{P}$ is positive definite

How can we solve this kind of problem?

How can we solve this kind of problem?

Could we perhaps use gradient descent?

Projected Gradient Descent

Quite like G.D., but we'll take a looong route there

Proximal Operator

It all begins with proximal operators:

Given a closed proper convex function:

$$f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$$

We define the (scaled) proximal operator $\mathbf{prox}_{\rho f}(x)$ as:

$$\mathbf{prox}_{\rho f}(x) = \underset{x'}{\operatorname{argmin}} \left(f(x') + \frac{1}{2\rho} \|x' - x\|_2^2 \right)$$

Intuitively, we search for a point x' that:

- Reduces the function value w.r.t. $f(x)$
- ...And is not too far from x

Proximal Point Method

The operator is the foundation for the proximal point method

Given a c.p.c. function f, we can minimizing via the simple iteration:

$$x^{(k+1)} = \mathbf{prox}_{\rho f}(x^{(k)})$$

The ρ parameter can also be updated, provided that:

- $\forall k : \rho_k > 0$
- $\sum_{k=1}^{\infty} \rho_k = \infty$

These are the same basic conditions for gradient descent convergence

Proximal Point Method

The operator is the foundation for the proximal point method

Given a c.p.c. function f, we can minimizing via the simple iteration:

$$x^{(k+1)} = \mathbf{prox}_{\rho f}(x^{(k)})$$

The ρ parameter can also be updated, provided that:

- $\forall k : \rho_k > 0$
- $\sum_{k=1}^{\infty} \rho_k = \infty$

These are the same basic conditions for gradient descent convergence

Pretty cool, right?

So, why haven't you (likely) ever heard about this?

The Most Useful Useless Algorithm

Because the proximal point method is (apparently) useless

Every iteration requires to solve:

$$\operatorname*{argmin}_{x'} \left(f(x) + \frac{1}{2\rho} \|x' - x\|_2^2 \right)$$

- This can be just as hard as minimizing $f(x)$ directly
- ...Or even more, give the quadratic term!

The Most Useful Useless Algorithm

Because the proximal point method is (apparently) useless

Every iteration requires to solve:

$$\text{argmin}_{x'} \left(f(x) + \frac{1}{2\rho} \|x' - x\|_2^2 \right)$$

- This can be just as hard as minimizing $f(x)$ directly
- ...Or even more, give the quadratic term!

...And yet it finds some suprising uses!

- It provides a form of regularization that enhances numerical stability
- It serves as a framework for studying many other algorithms
- ...And as a basis for deriving other algorithms

The Proximal Gradient Method

One such offspring is the Proximal Gradient Method

Let's consider a minimization problem in the form:

$$\underset{x}{\operatorname{argmin}}\ f(x) + g(x)$$

Where:

- f is c.p.c., has domain in $\mathbb{R}$, and is differentiable
- g is c.p.c., has domain in $\mathbb{R} \cup \{+\infty\}$, and is typically non differentiable

This type of problem is usually obtain by splitting an original cost function

- The friendlier part goes in f
- The nastier one in g

The Proximal Gradient Method

The problem with the split cost can then be solved by iterating:

$$x^{(k+1)} = \mathbf{prox}_{\rho_k g}\left(x^{(k)} - \rho_k \nabla f(x^{(k)})\right)$$

- First we perform a gradient descent step to minimize f
- ...Then one proximal step to minimize g (which is usually not differentiable)

The Proximal Gradient Method

The problem with the split cost can then be solved by iterating:

$$x^{(k+1)} = \mathbf{prox}_{\rho_k g}\left(x^{(k)} - \rho_k \nabla f(x^{(k)})\right)$$

- First we perform a gradient descent step to minimize f
- ...Then one proximal step to minimize g (which is usually not differentiable)

You can view the method as follows:

- We break a single proximal step in two sub-step
- The first substep applies to f and is implemented as a G.D. update
- In fact, G.D. updates can always be seen as proximal operator applications
- The second substep applies to g

If ρ_k is small enough, this approximates the original proximal point method

From Proximal to Projected Gradient Descent

This method is very useful when g represents the problem constraints

The feasible set C defined by a constraint can be associated to the function:

$$g_C(x) = \begin{cases} 0 \text{ if } x \in C \\ +\infty \text{ otherwise} \end{cases}$$

- This is called the indicator function of the constraint
- If the constraint defines a convex set, then g_C is also convex

From Proximal to Projected Gradient Descent

This method is very useful when g represents the problem constraints

The feasible set C defined by a constraint can be associated to the function:

$$g_C(x) = \begin{cases} 0 \text{ if } x \in C \\ +\infty \text{ otherwise} \end{cases}$$

- This is called the indicator function of the constraint
- If the constraint defines a convex set, then g_C is also convex

However, even a convex g is not differentiable!

- Hence, we cannot apply regular gradient descent
- ...But we can use the proximal gradient method!

From Proximal to Projected Gradient Descent

Let's inspect the proximal operator for g_C

By definition, this is given by:

$$\mathbf{prox}_{\rho g_C}(x) = \operatorname*{argmin}_{x'} \left(g_C(x') + \frac{1}{2\rho} \|x' - x\|_2^2 \right)$$

From Proximal to Projected Gradient Descent

Let's inspect the proximal operator for g_C

By definition, this is given by:

$$\mathbf{prox}_{\rho g_C}(x) = \operatorname*{argmin}_{x'} \left(g_C(x') + \frac{1}{2\rho} \|x' - x\|_2^2 \right)$$

- However, g_C is infinite on all infeasible points
- Therefore the operator reduces to:

$$\mathbf{prox}_{\rho g_C}(x) = \operatorname*{argmin}_{x' \in C} \|x' - x\|_2^2$$

Meaning that we find the point in C that is closest to x, i.e. we project x on C

From Proximal to Projected Gradient Descent

Let's consider the constraints we care about, i.e. $x \geq 0$

For these, we get:

$$\textbf{prox}_{\rho g_C}(x) = \underset{x' \geq 0}{\operatorname{argmin}} \|x' - x\|_2^2$$

Which can be decomposed into one problem for each $\boldsymbol{x}$ component:

$$\underset{x'_j \geq 0}{\operatorname{argmin}} \|x'_j - x_j\|_2^2$$

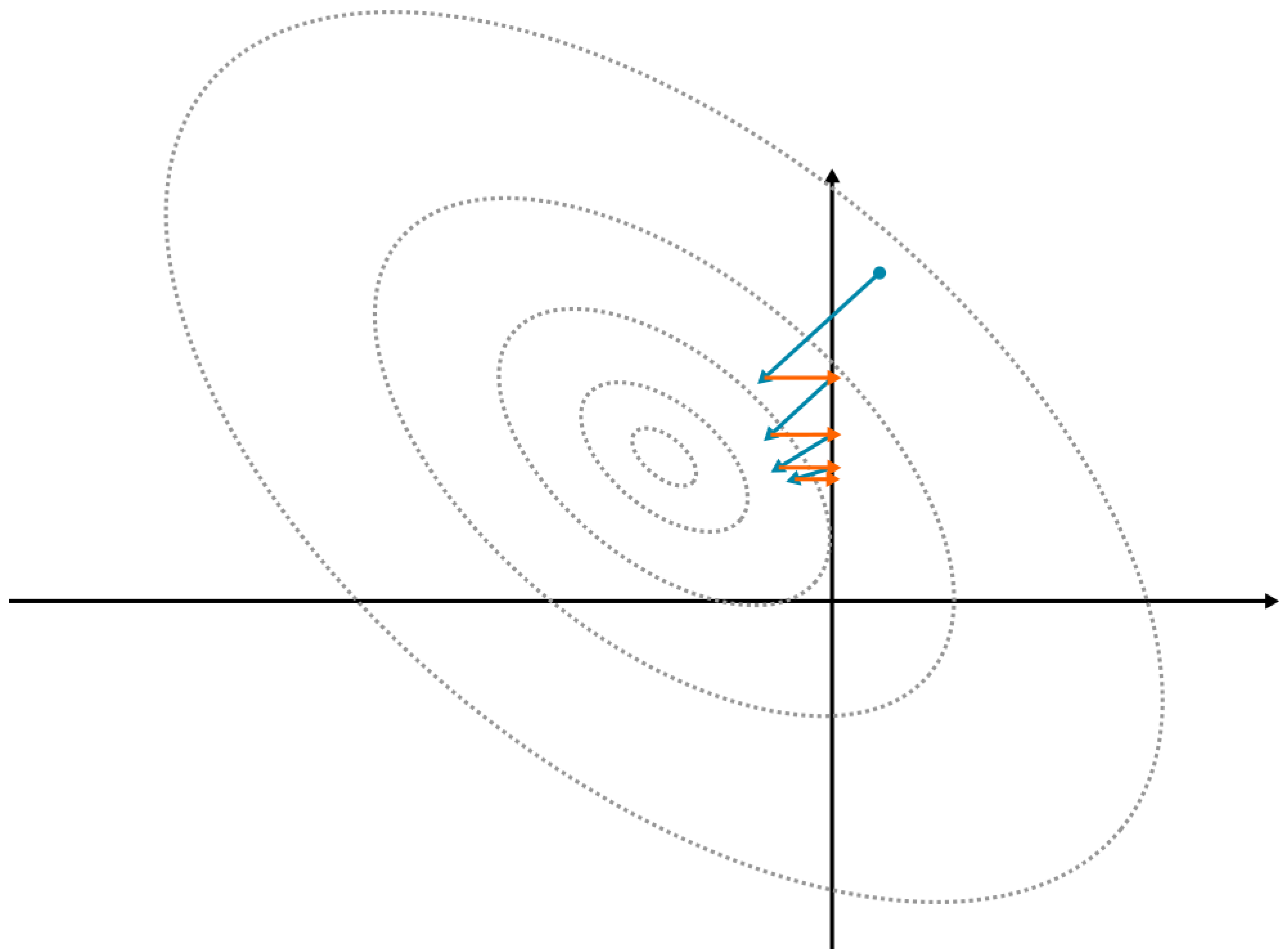
Which is just the same as performing a simple clipping:

$$\max(0, x_j)$$

So, in our case, projection is very easy!

Projected Gradient Descent

This is a visual intuition of how the method works



- The gradient steps bring the method closer to high quality solutions

Let's go back to our problem, now:

$$\underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} x^T P x + q^T x \mid x \geq 0 \right\}$$

How else could we tackle that?

The Interior Point Method

Actually, an interior point method

Constraints as Penalties

Let's pretend there were no constraints in out problem:

$$\underset{x}{\operatorname{argmin}} \ \frac{1}{2} x^T P x + q^T x$$

Would we really solve it via gradient descent?

Constraints as Penalties

Let's pretend there were no constraints in out problem:

$$\underset{x}{\operatorname{argmin}} \frac{1}{2} x^T P x + q^T x$$

Would we really solve it via gradient descent?

Since P is positive definite, the problem is (strongly) convex

...Which means there is a single optimal point, where the gradient is null

$$\nabla \left(\frac{1}{2} x^T P x + q^T x \right) = \frac{1}{2}(P + P^T)x + q = 0$$

Solving this linear system would give the optimal solution

From Constrained to Unconstrained Problems

Could we cast the constrained problem into a unconstrained one?

$$\underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} x^T P x + q^T x \mid x \geq 0 \right\}$$

We could achieve this by making sure that violating constriants incurs a cost

From Constrained to Unconstrained Problems

Could we cast the constrained problem into a unconstrained one?

$$\underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} x^T P x + q^T x \mid x \geq 0 \right\}$$

We could achieve this by making sure that violating constriants incurs a cost

For example, we can associate each $\mu_j \geq 0$ constraint to a penalty term:

$$x_j \geq 0 \longleftrightarrow -\log x_j$$

This particular type of penalty term is called a log barrier:

- When x_j approaches 0, the term approaches $+\infty$
- The barrier value is not defined for $x_j < 0$

From Constrained to Unconstrained Problems

The resulting unconstrained formulation is:

$$\underset{x}{\operatorname{argmin}} \frac{1}{2} x^T P x + q^T x - \mu \sum_{j=1}^{n} \log x_j$$

This formulation approximates the original one (since $x_j = 0$ is not permitted)

- As μ gets close to 0, we get a better approximation
- ...So that we can approximate the true solution with arbitrary precision

From Constrained to Unconstrained Problems

The resulting unconstrained formulation is:

$$\operatorname*{argmin}_x \frac{1}{2} x^T P x + q^T x - \mu \sum_{j=1}^{n} \log x_j$$

This formulation approximates the original one (since $x_j = 0$ is not permitted)

- As μ gets close to 0, we get a better approximation
- ...So that we can approximate the true solution with arbitrary precision

But how to we obtain a solution?

- Since $-\log x_j$ is a convex function, the cost is still strongly convex
- ...Which means we can still find the optimum by equating the gradient to 0

Solving Problems with Log Barriers

Equating the gradient to 0 gives us:

$$(P + P^T)x + q - \mu \frac{1}{x} = 0$$

- Where $1/x$ refers to the vector $\{1/x_j\}_{j=1}^{n}$

...Which is a system of non-linear equations

Solving Problems with Log Barriers

Equating the gradient to 0 gives us:

$$(P + P^T)x + q - \mu\frac{1}{x} = 0$$

- Where $1/x$ refers to the vector $\{1/x_j\}_{j=1}^n$

...Which is a system of non-linear equations

The system can be solved via the Newton-Raphson method

...Which is a second-order method for solving non-linear equations

- The method works by finding zeros of a local linear approximation
- It has very fast convergence...
- ...But it is prone to numerical issues that can jeopardize convergence

The Interior Point Method

So, to improve stability, it is common to rely on interior point methods

We consider a sequence of subproblems, each with a different μ_k:

$$\operatorname*{argmin}_x \frac{1}{2} x^T P x + q^T x - \mu_k \sum_{j=1}^{n} \log x_j$$

- The solution of each problem is a point inside the feasible region
- The sequence $\{\mu_k\}_{k=1}^{\infty}$ should be non-increasing
- ...So every problem will be a better approximation of the original one

The Interior Point Method

So, to improve stability, it is common to rely on interior point methods

We consider a sequence of subproblems, each with a different μ_k:

$$\operatorname*{argmin}_x \frac{1}{2} x^T P x + q^T x - \mu_k \sum_{j=1}^{n} \log x_j$$

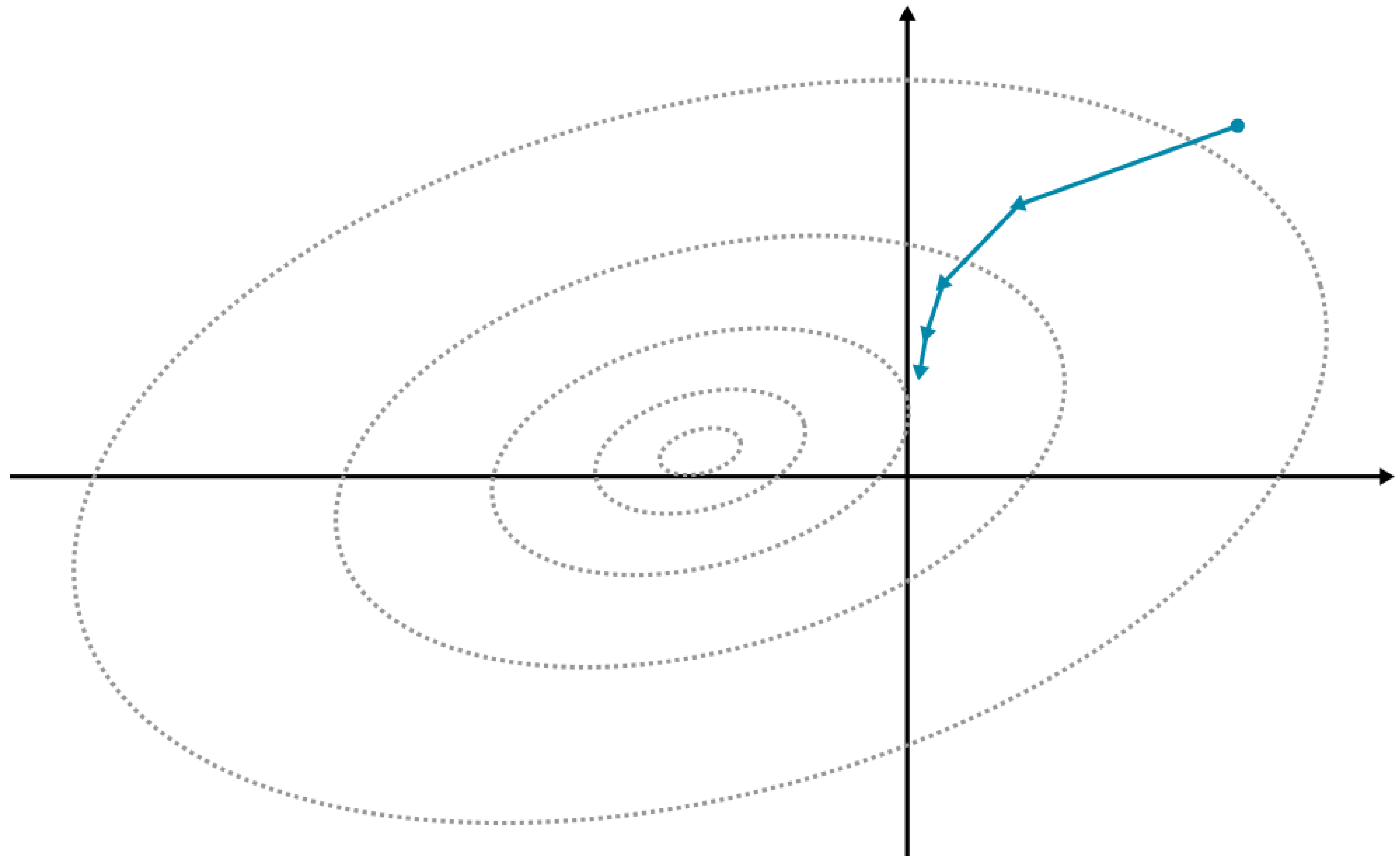
- The solution of each problem is a point inside the feasible region
- The sequence $\{\mu_k\}_{k=1}^{\infty}$ should be non-increasing
- ...So every problem will be a better approximation of the original one

The N.R. method is typically used to solve each subproblem

- Typically, the N.R. method is run just for one or a few iterations
- Under some (robust) conditions, this still enables convergence

The Interior Point Method

This is a visual intuition of how the method works



- At every iteration, we stay inside the feasible region
- Convergence can be very fast (and poly time, for some classes of problems)

More on Interior Point Methods

Interior Point Methods form a large family

- They can support more than simple non-negativity constraints
- E.g. linear constraints, other forms of convex constraints

Effective algorithm implementations are publicly available

More on Interior Point Methods

Interior Point Methods form a large family

- They can support more than simple non-negativity constraints
- E.g. linear constraints, other forms of convex constraints

Effective algorithm implementations are publicly available

One of currently fastest methods for QP is an interior point method

That is the PIQP solver developed by EPFL researchers

- PIQP actually combines an interior point method with a proximal point method
- The results is very fast and stable (but rather complex) solution method

This is the solver we are going to use in our implementation

Proximal and Interior Point methods are special

Because the showcase a key point

An Important Insight

If you look hard enough, you'll discover that there are are just two ways to deal with constraints

- You can covert constraints to penalties
- ...Or you can deal with them via projection

Both approaches attempt to simplify the original problem

An Important Insight

If you look hard enough, you'll discover that there are are just two ways to deal with constraints

- You can covert constraints to penalties
- ...Or you can deal with them via projection

Both approaches attempt to simplify the original problem

The penalty approach tries to make it closer to an unconstrained version

- If you go for this approach, you'll often deal with approximations
- You'll need to be sure that the approximation is good enough
- ...And also stable enough from a numerical point of view!

An Important Insight

If you look hard enough, you'll discover that there are are just two ways to deal with constraints

- You can covert constraints to penalties
- ...Or you can deal with them via projection

Both approaches attempt to simplify the original problem

The projection approach deals with constraints via a separate procedure

- If you go for this approach, you'll need to have one such procedure
- Convergence speed will typically be an issue
- ...But you can gain in precision and generality

Some References

Some references, in case you want to know more

- "Convex Optimization", by Boyd and Vandenberghe
- "Proximal Algorithms", by Parikh and Boyd
- "An interior-point proximal method of multipliers for convex quadratic programming", by Pougkakiotis and Gondzio