# Math Cheatsheet

## Notation

Given $x \in \mathbb{R}^n$, we note that there are $n$ decision variables.

Commonly used sets:

- $\mathbb{R}$ is the set of **real numbers**.
- $\mathbb{R}^{++}$ is the set of **positive real numbers**.

Given vector spaces $V$ and $W$ of dimensions $m$ and $n$ respectively:

- A function $f : V \to W$ is **linear** if $f(v) = Ab$ for some $n \times m$ matrix $A$. $(A \in R^{m \times n})$
- A function $f : V \to W$ is **affine** if $f(v) = Av + b$ for some matrix $A$ and vector $b$.

## Eigenvalues

Let $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ be a symmetric $2 \times 2$ matrix[1]:

- $A$ is a positive definite iff $D_k > 0$ for all leading principal minors:

$$D_1 = a$$
$$D_2 = ac - b^2$$

- $A$ is a positive semi-definite iff $\nabla_k \geq 0$ for all principal minors:

$$\nabla_1 = a$$
$$\nabla_1 = c$$
$$\nabla_2 = ac - b^2$$

## Derivatives

$$\frac{d}{dx} x^p = px^{p-1} \text{ for any } p \neq 0$$

$$\frac{d}{dx} e^x = e^x, \qquad \frac{d}{dx} \ln(x) = \frac{1}{x}$$

$$\frac{d}{dx} f(g(x)) = f'(g(x))g'(x)$$

$$\frac{d}{dx} f(x)g(x) = f'(x)g(x) + f(x)g'(x)$$

Consider $f(x) = \frac{1}{2} \parallel Ax - b \parallel^2$ for $A \in R^{m \times n}$:

$$\nabla f(x) = A^T(Ax - b)$$

Note that $\|x^2\| = \langle x, x \rangle = x^T x$.

Consider $f(x) = x^T Ax$ for $A \in R^{n \times n}$:

$$\nabla f(x) = (A^T + A)x$$

[1] definitions gathered from https://www.dr-eriksen.no/teaching/GRA6035/2010/lecture5.pdf

# Convexity & Convex Sets/Functions/Optimisation

## General optimisation

Optimisation problems are given in the form of:

$$\min_{x \in \mathbb{R}^n} f_0(x)$$

Optimisation variable

$f_0 : \mathbb{R}^n \to \mathbb{R}$
Objective / cost function
$f_0$ is not necessarily convex

subject to

$$f_{i(x)} \leq 0 \qquad i = 1, ..., m$$
$$h_{i(x)} = 0 \qquad i = 1, ..., p$$

$\mathbb{R}^n \to R$
Inequality constraint functions
Equality constraint functions

The optimal value is $p^* = \inf\{f_0(x) \mid f_{i(x)} \leq 0, h_{i(x)} = 0, \forall i\}$. Implementation varies on equation.

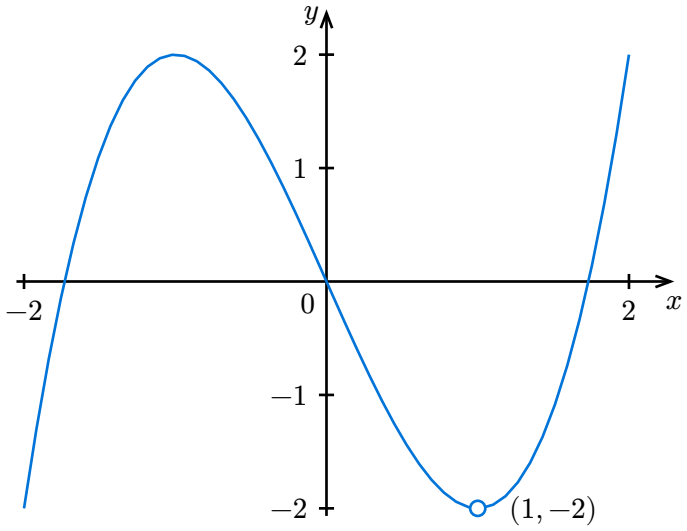
## Optimal and locally optimal points

$x$ is **feasible** if $x \in \text{dom } f_0$ and it satisfies the constraints.

A feasible $x$ is **optimal** if $f_0(x) = p^*$. $X_{\text{opt}}$ is the set of optimal points.

$x$ is **locally optimal** if there is an $\varepsilon > 0$ such that $x$ is optimal for $\min_z f_0(z)$ subject to:

$$f_i(x) \leq 0 \qquad i = 1, ..., m$$
$$h_i(x) = 0 \qquad i = 1, ..., p$$
$$\| z - x \|_2 \leq \varepsilon$$




Example 1: $f_0(x) = x^3 - 3x$

### Implicit constraints

INCOMPLETE

### Convex optimisation

Convex optimisation problems are given in the form of:

$$\min_{x \in \mathbb{R}^n} f_0(x)$$
$$\text{subject to}$$
$$f_i(x) \leq 0 \qquad i = 1, ..., m$$
$$h_i(x) = 0 \qquad i = 1, ..., p$$
$$\text{where}$$
$$f_0, f_1, ..., f_m \text{ must be convex}$$
$$h_1, ..., h_p \text{ must be affine}$$

A common form for $h_i = w^T x + b$.

The constraints form a convex set: if $x, y \in \mathbb{R}^n$ both satisfy all of the constraints, then so does any $\lambda x + (1 - \lambda) y$ for $\lambda \in [0, 1]$.

Every local optimum of a convex optimisation problem is a global optimum, i.e. given a local minium $x$ and a global minimum $y$, $f(x) > f(y)$.

# Gradient Methods & Stochastic Optimisation

## Gradient descent

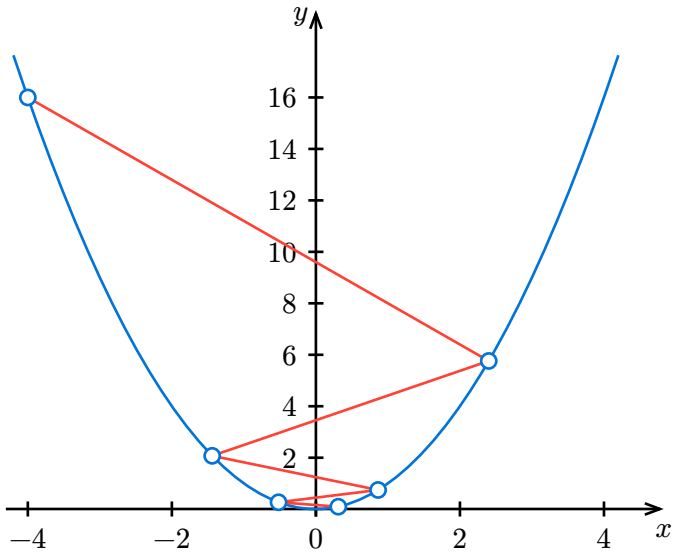
**Gradient descent** is a method to find local optima of a differentiable function $f$.
By intuition, the gradient tells us direction of greatest increase, and vice-versa for negative gradient.
So we take steps in directions that reduce function value.

The algorithm works as follows:

- Pick an initial point $x_0 \in \mathbb{R}^n$
- Repeat $x_{t+1} = x_t - \gamma_t \nabla f(x_t)$, for $t = 1, 2, 3, \ldots$

Where $\gamma_t$ is the $t^{\text{th}}$ step size / learning rate.

We must choose a **stopping criteria**, e.g. iterate until $\| \nabla f(x_t) \| \leq \varepsilon$ for $\varepsilon > 0$.




Example 2: $f(x) = x^2$, step size: 0.8

$$
\begin{aligned}
x^{(0)} &= 4 \\
x^{(1)} &= -4 - .8 \cdot 2 \cdot (-4) \\
x^{(n)} &= \ldots
\end{aligned}
$$

The step size we pick is important, as we may be in a situation where we are prone to overshoot and potentially end up in the wrong local minima:

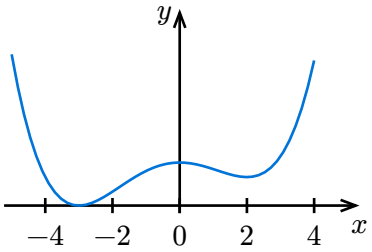



Figure 1: Step size matters.

INCOMPLETE gradient descent interpretation

## Backtracking line search

In **exact line search**, instead of picking a fixed step size that may or may not result in a decrease in function value, we consider minimising the function along the direction specified by the gradient to guarantee that the next iteration decreases the function value.

In other words, we choose

$$x_{t+1} \in \arg\min_{\gamma \geq 0} f(x_t - \gamma \nabla f(x_t))$$

This can be expensive to solve exactly, however if $f$ is convex, then this is a univariate convex optimisation problem.

With **backtracking line search**, we start with a large step size, $\gamma$, and keep shrinking it until

$$f(x_t - \gamma \nabla f(x_t)) < f(x_t) - C$$

where $C$ is an 'acceptable threshold'

This always guarantees a decrease, but it may not decrease as much as line search.
(though in practice this isn't a huge issue as it's still typically faster)

**hyper-parameters**[2]: $\gamma_{\text{init}}$, $0 < \beta < 1$ and $0 < \alpha < \frac{1}{2}$

**for each iteration** ▷ the stopping criteria is omitted here for simplicity
  $\gamma \leftarrow \gamma_{\text{init}}$
  **while** $f(x_t - \gamma \nabla f(x_t)) > f(x_t) - \alpha \cdot \gamma \cdot \| \nabla f(x_t) \|_2^2$ **then**
    $\gamma \leftarrow \beta\gamma$ ▷ shrink step size
  **end**
  $x_{t+1} \leftarrow x_t - \gamma \nabla f(x_t)$ ▷ gradient descent update
**end**

Algorithm 1: Gradient Descent w/ Backtracking Line Search

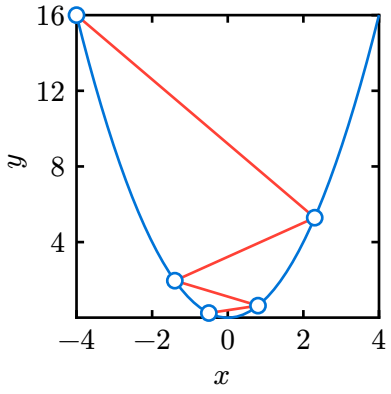



Figure 2: $\alpha = 0.2, \beta = 0.99$

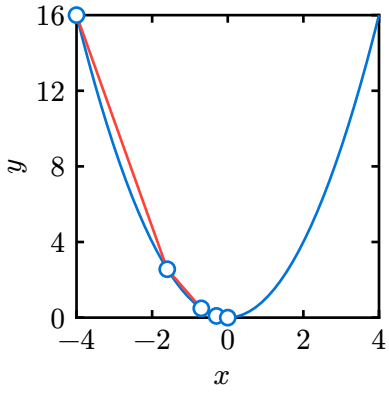



Figure 3: $\alpha = 0.1, \beta = 0.3$

[2] **hyper-parameters** are parameters who values affect the learning process.

## Optimality citerion

An **unconstrained problem** does not include any equality or inequality constraints.

**Example: $x$ is optimal iff $x \in \text{dom } f_0, \nabla f_0(x) = 0$.**

$x$ is optimal iff it is feasible and $\nabla f_0(x)^T (y - x) \geq 0$ for all feasible $y$

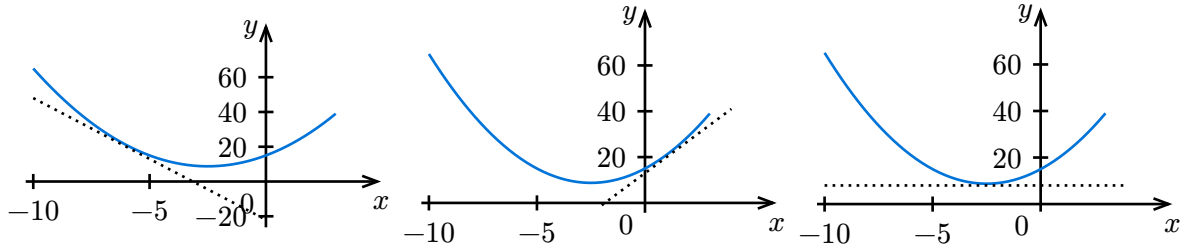
*Proof*:

$$f_0(y) \geq f_0(x) + \nabla f_0(x)^T (y - x)$$
$$f_0(y) - f_0(x) \geq \nabla f_0(x)^T (y - x)$$
$$f_0(y) - f_0(x) \geq 0 \text{ for all feasible } y$$

Thus $x$ is optimal as $f_0(x)$ is smaller than any $f_0(y)$. thm-qed-symbol

## Gradients of Convex Functions

For a differentiable convex function $g(x)$, its gradients yield **linear underestimators**. A zero gradient corresponds to a global optimum.




## Subgradients

A **subgradient** of a convex $f$ at $x$ is any $g \in \mathbb{R}^n$ such that $f(y) \geq f(x) + g^T(y - x)$ for all $y$.

- Always exists for convex functions.
- If $f$ differentiable at $x$, then $g = \nabla f(x)$ uniquely.
- For non-convex $f$, we use same definition, but subgradients need not exist.

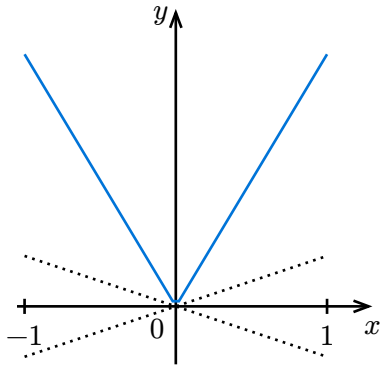



Figure 4: $f : \mathbb{R} \to \mathbb{R}, f(x) = |x|$
For $x \neq 0$, unique subgradient is $g = \text{sign}(x)$.
For $x = 0$, subgradient $g$ is any element of $[-1, 1]$.

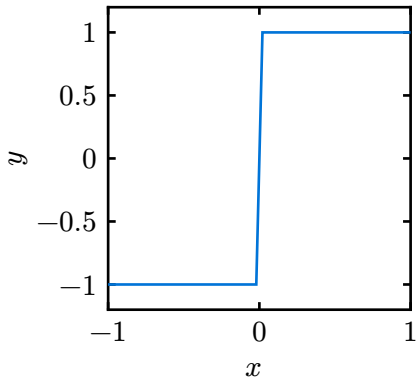



Figure 5: $\partial f(x)$

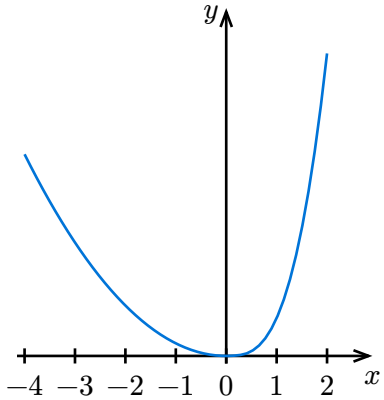



Figure 6: $f(x) = \max(f_1(x), f_2(x))$
for $f_1$, $f_2$ convex, differentiable

If $f_1(x) > f_2(x)$, subgradient $g = \nabla f_1(x)$.
If $f_2(x) > f_1(x)$, subgradient g = nabla f_2(x).
If f_1(x) = f_2(x), then $\nabla f_1(x)$ and $\nabla f_2(x)$ are both subgradients (and so are all convex combinations).

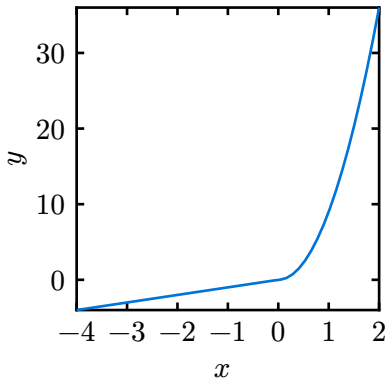



Figure 7: $\partial f(x)$

## Subgradient Descent

Consider $f$ convex, having $\text{dom}(f) = \mathbb{R}^n$ but not necessarily differentiable. With the **subgradient method**, we perform a similar routine to gradient descent, but replacing gradients with subgradients:

1. Pick an initial point $x_0 \in \mathbb{R}^n$
2. Repeat $x_{t+1} = x_t - \gamma_t s_{f(x_t)}$ for $t = 1, 2, 3, \ldots$

Where $\gamma_t$ is the $t^{\text{th}}$ step size, and $s_{f(x_t)}$ is a subgradient of $f$ at $x_t$.

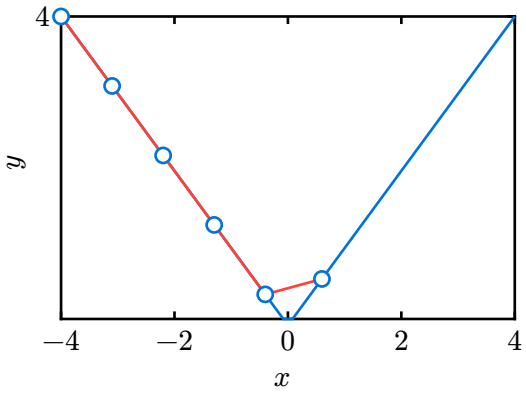



Figure 8: $\gamma = 0.9$

A fixed step size may not result in convergence for non-differentiable functions.
Instead, we can use a **diminishing step size** which has the property that step size must decrease as number of iterations increaes but not too quickly that the algorithm fails to make progress.

Common diminishing step size rules:

- $y_t = \frac{a}{b+t}$ for some $a > 0, b \geq 0$
- $y_t = \frac{a}{\sqrt{t}}$ for some $a > 0$

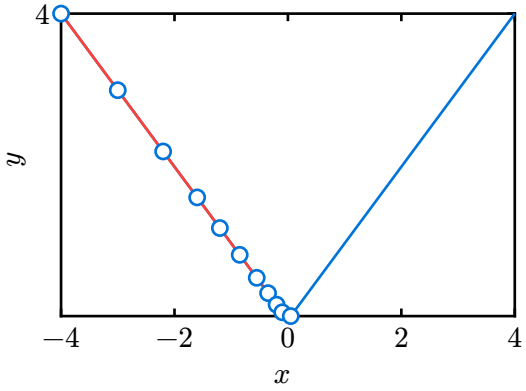



Figure 9: Diminishing step size

The hard work in convex optimisation is to identify which conditions guarantee quick convergence within a small error of the optimum.

Let $f_{\text{best}}^{(t)} = \min_{t' \in \{0,\ldots,t\}} f(x_{t'})$

For a fixed step size, $\gamma$, we are guaranteed that

$$\lim_{t \to \infty} f_{\text{best}}^{(t)} - \inf_x f(x) \leq \varepsilon(\gamma)$$

where $\varepsilon(\gamma)$ is some positive constant that depends on $\gamma$.

If $f$ is differentiable, then we have $\varepsilon(\gamma) = 0$ whenever $\gamma$ is small enough.

## Stochastic gradient descent

Consider minimising the average of functions

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

As $\nabla \sum_{i=1}^{m} f_i(x) = \sum_{i=1}^{m} \nabla f_i(x)$, gradient descent would repeat:

$$x_{t+1} = x_t - \gamma_t \sum_{i=1}^{m} \nabla f_i(x_t) \quad \text{for } t = 1, 2, 3, \ldots$$

In the **stochastic gradient descent** algorithm, we instead repeat:

$$x_{t+1} = x_t - \gamma_t \nabla f_{i_t}(x_t) \quad \text{for } t = 1, 2, 3, \ldots$$

where $i_t \in \{1, \ldots, m\}$ is some chosen index at iteration $t$.

SGD is appealing as:
- The iteration cost is independent of $m$ (no. of functions)
- Can save on memory usage

There are two rules for choosing the index $i_t$:
- **Cyclic rule**: choose $i_t = 1, 2, \ldots, m, 1, 2, \ldots, m, \ldots$
- **Randomised rule**: choose $i_t \in \{1, \ldots, m\}$ uniformly at random

  More common in practice. Note that, $E\left[\nabla f_{i_t}(x)\right] = \nabla f(x)$.

  As such, we can view SGD as an unbiased estimate of gradient at each step.

**Example of stochastic regression**

Given data points $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^d$ and $y^{(1)}, \ldots, y^{(m)} \in \mathbb{R}$, find the best fit polynomial of degree $d$.

$$\min_{c \in \mathbb{R}} \sum_{i=1}^{m} \left(y^{(i)} - c^T x^{(i)}\right)^2$$

Gradient computation $\nabla f(c) = \frac{1}{m} \sum_{i=1}^{m} 2\left(y^{(i)} - c^T x^{(i)}\right) x^{(i)}$ is doable when $m$ is moderate, but not when $m$ is huge.

It is clear that 10k stochastic steps is much more afforable when:
- one batch (full gradient) update costs $O(md)$
- one stochastic update costs $O(d)$

# Constrained Optimisation

Suppose we want to optimise

$$\min_x f(x)$$

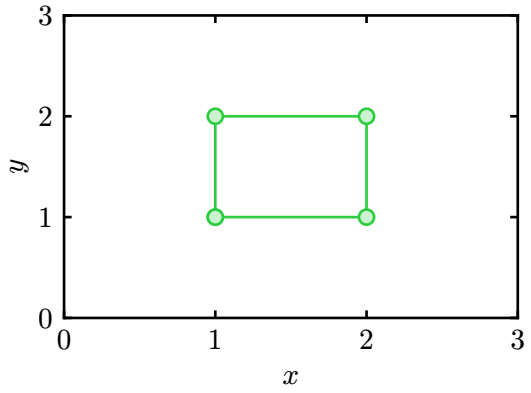
$$\text{subject to } x \in C$$

Where $f(\cdot)$ is a convex function and $C \subseteq \mathbb{R}^n$.

**Example: minimise $f(x, y)$ subject to $x \in [a, b]$ and $y \in [c, d]$**

Much more challenging than univariate case.

The optima can occur at a critical point, $\nabla f = 0$, or anywhere on the boundary of the rectangle determined by $(1, 1), (2, 1), (1, 2), (2, 2)$.




## Projected Gradient Descent

The **projected gradient descent algoritm** works as follows:

- Pick an initial point $x_0 \in C$
- Repeat $x_{t+1} = \text{proj}_{C(x_t - \gamma_t \nabla f_0(x_t))}$ for $t = 1, 2, 3, \ldots$

Where $\gamma_t$ is the $t^{\text{th}}$ step size and $\text{proj}_{c(y)}$ is a Euclidean projection of $y$ onto the set $C$.

$$\text{proj}_{c(z)} = \arg\min_{x \in C} \| x - z \|_2^2$$

Given a set $C \subseteq \mathbb{R}^n$, the projection of a point $z \in \mathbb{R}^n$ onto $C$ is the point $x \in C$ such that the distance between $x$ and $z$ is less than or equal to the distance between $z$ and any other $x' \in C$.

INCOMPLETE missing diagram

PGD provides the following advantages:

- Only one additional step on top of gradient descent
- Fast for simple constraints, e.g. $x \geq 0$
- Essentially same convergence guarantees as gradient descent if you can compute the projection

It has some disadvantages:

- Can be expensive if every step takes you outside of the set of constraints
- Can be expensive if constraint set is complicated
- Inaccuracy of the projection computation can cause convergence issues

### Examples of Projections

We may also project under different notions of distance.

- Project on to $C = \{x : x \leq 0\}$

$$\text{proj}_c(x)_i = \begin{cases} x, & \text{if } x_i \leq 0 \\ 0, & \text{otherwise} \end{cases}$$

- Project on to $C = \{x : \|x\|_2 \leq 1\}$

$$\text{proj}_c(x) = \begin{cases} x, & \text{if } \|x\|_2 \leq 1 \\ \frac{x}{\|x\|_2}, & \text{otherwise} \end{cases}$$

### Example PGD

**Example:** Projected gradient descent on
$f(x, y) = (x-2)^2 - (x-2)(y-2) + \frac{1}{2}(y-2)^2$

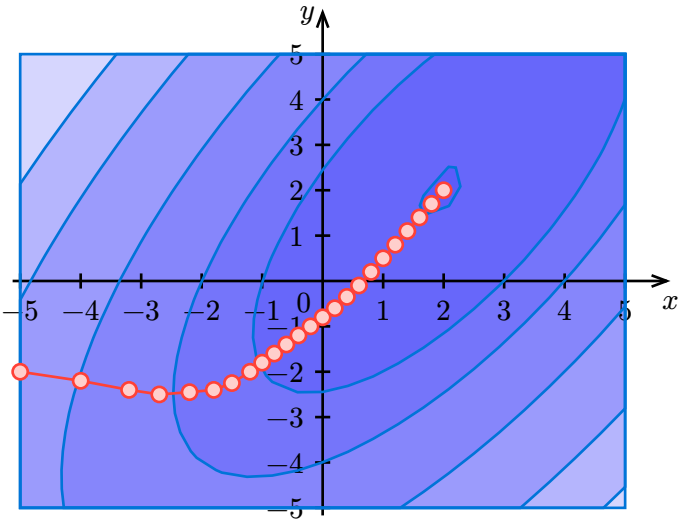



Figure 10: PGD starting at $(-5, -2)$, $\gamma = 0.1$

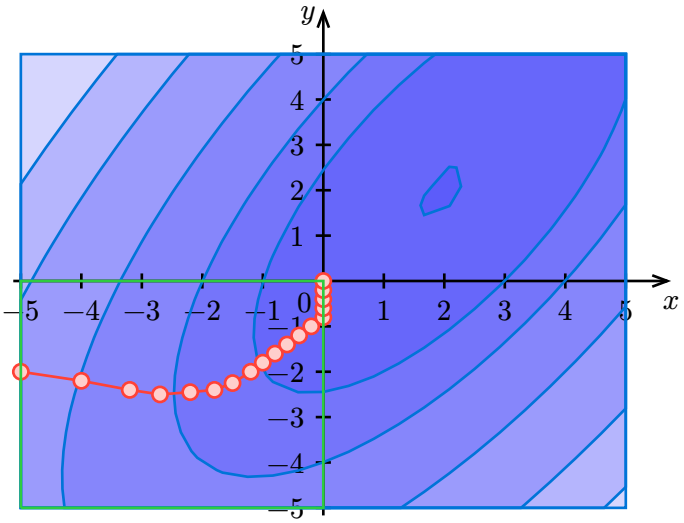



Figure 11: PGD starting at $(-5, -2)$, $\gamma = 0.1$, constraint $x \leq 0, y \leq 0$
The critical point is not feasible!

## Feasible Direction Methods

Generate a feasible sequence $\{x^t\} \subseteq C$ with iterations

$$x_t = x_t + \gamma_t d^t$$

where $d^t$ is a feasible direction (s.t. $x_t + \gamma_t d^t \in C$)

### Franke-Wolfe Algorithm

The **Frank-Wolfe algorithm** is an alternative to PGD.
To minimise a convex function over a convex set, it is sufficient to solve a series of linear optimisation problems. Specifically, approximate $f_0$ by its first order Taylor expansion at current estimate, and minimise this function over the set of constraints.
Then, take a step towards this new optimum.

Formally, $f_0(x) \approx f_0(x_t) + \nabla f_0(x_t)^T (x - x_t)$.

- Compute $s_t \in \arg\min_{x \in C} f_0(x_t) + \nabla f_0(x_t)^T$ (x - x_t)
- Set $x_{t+1} = x_t + \alpha(s_t - x_t)$

INCOMPLETE no diagram

The algorithm works as follows:

- Pick an initial point $x_0 \in C$
- Repeat:
  - $s_t = \arg\min_{x \in C} \langle x, \nabla f_0(x_t) \rangle$
  - $x_{t+1} = (1 - \gamma_t) x_t + \gamma_t s_t$

Note: there is no projection, update solved directly over $C$.

The default step size is $\gamma_t = \frac{2}{2+t}, t = 0, 1, 2, \ldots$.

Note for any $0 \leq \gamma_t \leq 1$, we have $x_{t+1} \in C$ by convexity. So we can write the update as:

$$x_{t+1} = x_t + \gamma_t (s_t - x_t)$$

INCOMPLETE skipped all example diagrams

INCOMPLETE stopping criteria

INCOMPLETE adv / disadv

## Duality and Lagrange Multipliers

### Lagrangian

Consider general minimisation problem $\min_{x \in \mathbb{R}^n} f_0(x)$ subject to:

$$\begin{aligned} f_i(x) &< 0, \qquad i = 1, ..., m \\ h_i(x) &= 0, \qquad i = 1, ..., p \end{aligned}$$

(does not need to be convex, but we pay attention to convex case)

We define **Lagrangian** as:

$$L(x, \lambda, v) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_{i(x)} + \sum_{i=1}^{p} v_i h_{i(x)}$$

- Incorporate constraints into a new objective function
- $\lambda_i$ is Lagrange multiplier associated with $f_{i(x)} \leq 0$, with $\lambda_i \geq 0$
- $v_i$ is Lagrange multiplier associated with $h_{i(x)} = 0$
- Lagrange multipliers can be thought of as enforcing soft constraints

**Example Lagrangian**

$$\max_{x,y \in \mathbb{R}} xy$$

$$\text{subject to}$$

$$x + y = 1$$

$$L(x, y, v_1) = -xy + v_1 \cdot (1 - x - y)$$

Given the Lagrangian above, we can solve as follows:

$$\begin{aligned} \frac{\partial L}{x} &= -y - v_1 \quad = 0 \\ \frac{\partial L}{y} &= -x - v_1 \quad = 0 \\ \frac{\partial L}{v_1} &= 1 - x - y = 0 \end{aligned}$$

We have $x = y$, $x + y = 1$, thus $x = x = 0.5$ is the only critical point of $L$.

### Lagrange multiplier

$$\min_{x \in \mathbb{R}^n} f_0(x)$$

$$\text{subject to}$$

$$h_i(x) = 0 \qquad i = 1, ..., p$$

**Theorem 0.1** (Lagrange multiplier theorem): if $x^*$ is a local minimum of the constrained optimisation problem and $\nabla h_1(x^*), ..., \nabla h_p(x^*)$ are linearly independent, then there exists a $v^* \in \mathbb{R}^p$ such that $\nabla L(x^*, v^*) = 0$.

This is also called the **method of Lagrange multipliers**, it is a strategy for finding local maxima and minima of a function subject to equality constraints.

Theorem 1: Lagrange multiplier theorem

**Lagrangian dual function**

The **Lagrange dual function** $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as:

$$\begin{aligned} g(\lambda, v) &= \inf_x L(x, \lambda, v) \\ &= \inf_x \left( f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x) \right) \end{aligned}$$

We construct a dual function by minimising the Lagrangian over the primal variables.

$$g(\lambda, v) = -\infty \text{ for some } \lambda \text{ and } v$$

**Theorem 0.2** (lower bound property): If $\lambda \succcurlyeq 0$, then $g(\lambda, v) \leq p^*$.

Theorem 2: lower bound property

*Proof*: If $\tilde{x}$ is feasible and $\lambda \succcurlyeq 0$ then

$$\begin{aligned} f_0(\tilde{x}) &\geq L(\tilde{x}, \lambda, v) \\ &\geq \inf_x L(x, \lambda, v) \\ &= g(\lambda, v) \end{aligned}$$

Minimising over all feasible $\tilde{x}$ gives $p^* \geq g(\lambda, v)$.

Recall that $p^*$ is the optimal value of the primal problem. thm-qed-symbol

### Lagrange dual problem

The **Lagrange dual problem** is defined as

$$\begin{gathered}\max_{\lambda,v} g(\lambda, v) \\ \text{subject to} \\ \lambda \geq 0\end{gathered}$$

- It finds the best lower bound on $p^*$, obtained from Lagrange dual function.
- It is a convex optimisation problem; optimal value denoted $d^*$.
- $\lambda, v$ are dual feasible if $\lambda \succcurlyeq 0, (\lambda, v) \in \text{dom } g$
- often simplified by making implicit constraint $(\lambda, v) \in \text{dom } g$ explicit

**Example: Standard form LP and its dual**

$$\begin{array}{ll}\min c^T x & \max -b^T v \\ \text{subject to} & \text{subject to} \\ Ax = b, x \geq 0 & A^T v + c \succcurlyeq 0\end{array}$$

### Weak and strong duality

**Weak Duality** means $d^* \leq p^*$

- Always holds (for convex/non-convex problems)
- Can be used to find non-trivial lower bounds for difficult problems.

**Example: Solving SDP to find lower bound for two-way partitioning problem**

$$\begin{gathered}\max -1^T v \\ \text{subject to} \\ W + \text{diag}(v) \succcurlyeq 0\end{gathered}$$

**Strong Duality** means $d^* = p^*$

- Does not generally hold
- Usually holds for convex problems
- Conditions that guarantee strong duality in convex problems are called **constraint qualifications**

**Duality gap**: given primal feasible $x$ and dual feasible $\lambda, v$, the quantity

$$f(x) - g(\lambda, v)$$

is called the duality gap between $x$ and $\lambda, v$.

Note that

$$f(x) - f(x^*) \leq \overbrace{f(x) - g(\lambda, v)}^{\text{Duality Gap}}$$

so if the **duality gap is zero**, then $x$ is **primal optimal** (and similarly, $\gamma, v$ are dual optimal).

This also provides a stopping criterion: if $f(x) - g(\lambda, v) \leq \varepsilon$ then we are guaranteed that $f(x) - f(x^*) \leq \varepsilon$. This is useful in conjunction with iterative methods.

**Slater's Condition**

For any optimisation problem of the form:

$$\min_{x \in \mathbb{R}^n} f_0(x)$$
$$\text{subject to}$$
$$f_i(x) \leq 0 \qquad i = 1, ..., m$$
$$h_i(x) = 0 \qquad i = 1, ..., p$$

Where $f_0, ..., f_m$ are *convex functions*, strong duality holds if there exists an $x$ such that:

$$f_i(x) < 0 \qquad i = 1, ..., m$$
$$h_i(x) = 0 \qquad i = 1, ..., p$$

**Complementary Slackness**

Suppose that there is a *zero duality gap*.

Let $x^*$ be an optimum of the primal and $(\lambda^*, v^*)$ be an optimum of the dual.

$$\begin{aligned} f_0(x^*) &= g(\lambda^*, v^*) \\ &= \inf_x \left[ f_0(x) + \sum_{i=1}^m \lambda_i^* f_i(x) + \sum_{i=1}^p v_i^* h_i(x) \right] \\ &\leq f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{i=1}^p v_i^* h_i(x^*) \\ &= f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) \\ &\leq f_0(x^*) \end{aligned}$$

So we find that:

$$\sum_{i=1}^m \lambda_i^* f_i(x^*)$$

As $\lambda \geq 0$ and $f_i(x_i^*) \leq 0$, this can only happen if $\lambda^* f_i(x^*) = 0$ for all $i$.

- If $f_i(x^*) < 0$ then $\lambda_i^* = 0$
- If $\lambda_i^* > 0$ then $f_i(x^*) = 0$

**KKT conditions**

The **Karush-Kuhn-Tucker (KKT) Conditions** for differentiable $f_i, h_i$:

- *primal constraints*: for all $i$, $f_i(x^*) \leq 0$ for all $i$, $h_i(x^*) = 0$
- *dual constraints*: for all $i$, $\lambda_i^* \geq 0$
- *complementary slackness*: for all $i$, $\lambda_i^* f_i(x^*) = 0$, $i = 1, ..., m$
- *gradient of Lagrangian*: with respect to $x$ vanishes:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, v^*) &= \nabla f_0(x) + \sum_{i=1}^m \lambda_i \nabla f_i(x) + \sum_{i=1}^p v_i \nabla h_i(x) \\ &= 0 \end{aligned}$$

If strong duality holds and $x, \lambda, v$ are optimal, then they must satisfy the KKT conditions.

INCOMPLETE proof

INCOMPLETE example quadratic mini

INCOMPLETE convex duality in practice