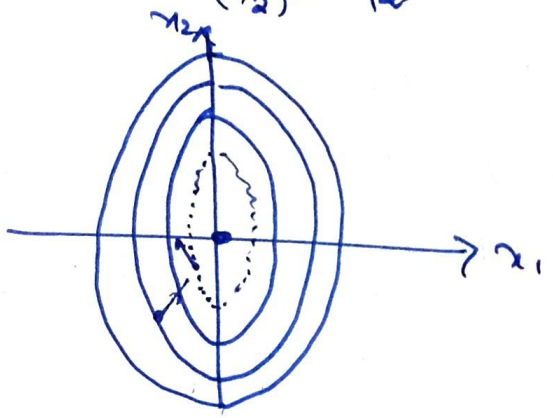
Conjugate Gradient Method

$$x_{k+1} = x_k + \alpha_k p_k .$$

Here, we choose p_k to be one of the coordinate directions. Then we solve an one-dimensional problem

$$\min f(x_k + \alpha p_k) \; ; \text{ where } p_k = e_j \text{ for some } j .$$

Example: $\min f(x_1, x_2) = 4x_1^2 + x_2^2 = \frac{x_1^2}{(1/2)^2} + \frac{x_2^2}{1^2}$



Let $x^{(1)} = (-1,-1)$ and choose
$p^{(1)} = (1,0)$. $[f(x^{(1)}) = 5$

$\Rightarrow x^{(2)} = (-1+\alpha, -1)$

and $f(x^{(2)}) = 4(\alpha-1)^2 + 1$
$= 4\alpha^2 - 8\alpha + 5$

$\Rightarrow \frac{\partial f}{\partial \alpha} = 8\alpha - 8 = 0 \Rightarrow \alpha = 1$.

$\Rightarrow x^{(2)} = (0,-1)$ and $f(x^{(2)}) = 1$.

Let $p^{(2)} = (0,1)$.

$\Rightarrow x^{(3)} = (0, -1+\alpha)$.

$f(x^{(3)}) = (\alpha-1)^2 \Rightarrow \frac{\partial f}{\partial \alpha} = \alpha - 1 = 0 \Rightarrow \alpha = 1$

Hence, $x^{(3)} = (0,0)$ and $f(x^{(3)}) = 0$.

Algorithm:

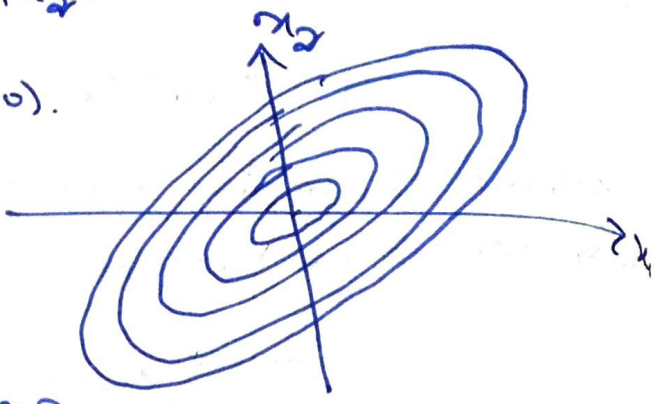
Step 1°. Choose p_k as one of the coordinates.

Step 2°. α : min $f(x_k + \alpha p_k)$

~~Do until~~

Do until $\|\nabla f\| \le \varepsilon$.

Example: $f(x) = 4x_1^2 + x_2^2 - 2x_1x_2$.



Let $x^{(1)} = (-1,-1)$ and choose $p^{(1)} = (1,0)$.

$\Rightarrow x^{(2)} = (-1+\alpha, -1)$ and

$f(x^{(2)}) = 4(\alpha-1)^2 + 1 - 2(\alpha-1)(-1)$

$= 4$

$\frac{\partial f}{\partial \alpha} = 8(\alpha-1) + 2 = 0 \Rightarrow \alpha = \frac{3}{4}$.

$\Rightarrow x^{(2)} = \left(-\frac{1}{4}, -1\right)$.

Let $p^{(2)} = (0,1)$. Then $x^{(3)} = \left(-\frac{1}{4}, -1+\alpha\right)$.

$\Rightarrow f(x^{(3)}) = 4\left(-\frac{1}{4}\right)^2 + (\alpha-1)^2 - 2\left(-\frac{1}{4}\right)(-1+\alpha)$

$\Rightarrow \frac{\partial f}{\partial \alpha} = 2(\alpha-1) + \frac{1}{2}(\alpha+1) = 0$

$\Rightarrow \alpha = \frac{3}{4}$.

$\Rightarrow x^{(4)} = \left(-\frac{1}{4}, -\frac{1}{4}\right)$

Example: $f(x) = \frac{1}{2}x^T H x + c^T x + d$.

For the first example,

$$H = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix}$$

and for second example,

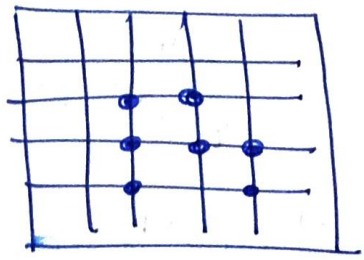
$$H = \begin{bmatrix} 8 & -2 \\ -2 & 2 \end{bmatrix}.$$

So unlike in example 1, we do not have minimum even after two steps in example 2. This is because the Hessian H is non diagonal. One solution is change of coordinates.

Remark° In a quadratic problem, it is guaranteed that the method converges in n steps ($n = \dim(x)$).

This is however not the same in case of non quadratic problems.

$$\frac{\partial \vec{u}}{\partial t} = f(u) \xrightarrow{\text{Finite difference method}}$$



u = weighted sum of basis functions

let $(x, y, t) \in$ Domain, $t \in [0, T]$

Solve $Ax = b$ (where $A \in \mathbb{R}^{n_g \times n_g}$)

and n_g = #grid points.

$$\text{Error} = \frac{1}{2} \| Ax - b \|^2_{\text{grid}, L^2} = f(x).$$

We now find $\min_x f(x)$. $\left[\nabla f = \frac{1}{2}(Ax-b)^T(Ax-b) \right]$

↓

This can be solved using conjugate method

Loss function

Read°. • Numerical methods to solve a differential equation system

• Newton forward, backward and central methods.

• Most PDE systems solve $f(x) = \frac{1}{2}\|Ax - b\|^2$ problems by converting it to an optimization problem.

Consider the problem,

$$\min f(x) = \frac{1}{2} x^T H x + c^T x,$$

where H = Hessian.

Let $x = x^{(0)} + \sum_{i=0}^{n-1} \alpha^{(i)} p^{(i)}$.

Suppose $\{p^{(i)}\}_i$ and $x^{(0)}$ are given. Then we need to find $\alpha^{(i)}$

$$\psi(\alpha) := \frac{1}{2}\left(x^{(0)} + \underbrace{\alpha^{(i)} p^{(i)}}_{\sum_i \alpha^{(i)} p^{(i)}}\right)^T H\left(x^{(0)} + \alpha^{(i)} p^{(i)}\right) + c^T\left(x_0 + \alpha^{(i)} p^{(i)}\right)$$

(Einstein's notation)

Let $P = \left[p^0 \mid p^1 \mid \cdots \mid p^{n-1}\right]$ and $\alpha = \left[\alpha^{(0)} \ldots \alpha^{(n-1)}\right]^t$.

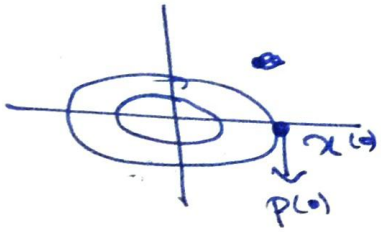
$$\Rightarrow \psi(\alpha) = \frac{1}{2}\alpha^T P^T H P \alpha + (Hx^{(0)} + c)^T P\alpha + \frac{1}{2} x^{(0)} H x^{(0)} + c^T x^{(0)}.$$

We want the matrix PHP a diagonal matrix.

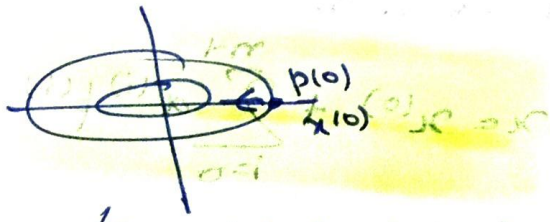
i.e. $(P^T H P)_{ij} = P_i^T H P_j = \begin{cases} = 0, & i \neq j \\ \neq 0, & i = j. \end{cases}$

Remark: If $P_i^T H P_j = 0 \ \forall i \neq j$, the vectors $\{P_j\}$ are called H-conjugate vectors.

- Given H, does a set of H conjugate vectors exist?
- If yes, how to find it?
- If the problem we are solving is a convex quadratic form, H is always symmetric and hence orthogonally diagonalisable.
- A convex ~~con~~ quadratic function can be minimized in atmost n steps (using the eigendirections).



(Not solvable in 2 steps)



(Solvable in 2 steps)

Recall: We want to minimize $f(x) = \frac{1}{2} x^T H x + c^T x$ and

$$P_i H P_j \begin{cases} = 0, & i \neq j \\ \neq 0, & i = j \end{cases}$$

$\Rightarrow \nabla f = Hx + c = 0 \Rightarrow x = x^*$ is the optimal solution.

For the kth iteration, we consider the residual vector

$$\begin{aligned} r_k &= Hx_k - Hx^* \\ &= Hx_k + c \\ &= \nabla f(x_k). \end{aligned}$$

Remark: For steepest descent, the descent direction r_k is the direction of gradient at that point (for convex optimization problem). $\boxed{x_{k+1} = x_k + \alpha_k p_k = x_k - \alpha_k r_k.}$

- For conjugate gradient,

$$\boxed{x_{k+1} = x_k + \alpha_k [-r_k + \beta_k p_{k-1}]}$$

$p_{k-1}^T H p_{k-1} \neq 0$ and we choose $p_k = -r_k + \beta_k p_{k-1}$ and require that $p_{k-1}^T H p_k = 0$.

↳ Descent direction in C.G. ↓

(~~H conjugate~~)

We want to get hold of H conjugate directions

Now,

let $\phi(\alpha) = \frac{1}{2}(x_k + \alpha_k p_k)^T H (x_k + \alpha_k p_k) + c^T (x_k + \alpha_k p_k)$

$= \frac{1}{2} \alpha_k p_k^T H p_k \alpha_k + c^T \alpha_k p_k + \cdots$ (~~some constant~~) $+ f(x_k)$

$$\Rightarrow \frac{\partial \phi}{\partial \alpha} = - \frac{c^T p_k - p_k^T H p_k}{p_k^T H p_k}$$

$$= \frac{-(c + Hp_k)^T p_k}{p_k^T H p_k}$$

$$= - \frac{r_k^T p_k}{p_k^T H p_k}$$

$$\left[p_{k-1}^T H p_k = 0 \Rightarrow p_{k-1}^T H (-r_k + \beta_k p_{k-1}) = 0 \Rightarrow \beta_k = \frac{p_{k-1}^T H r_k}{p_{k-1}^T H p_{k-1}} \right]$$

Now, for

$$P_{k-1}^T H P_k = 0$$

$$\Rightarrow P_{k-1}^T H (-r_k + \beta_k P_{k-1}) = 0$$

$$\Rightarrow \beta_k = \frac{P_{k-1}^T H r_k}{P_{k-1}^T H P_{k-1}}$$

Remark: Finding eigendirections is expensive. So, we start with a desirable direction and choose the subsequent directions in such a way that $\{P_j\}$ forms a set of H-conjugate vectors.

Conjugate Direction Algorithm

Step 1: Given initial guess x_0, choose $P_0 = -r_0 = -(Hx_0 + c)$.

Step 2: ~~while ($\|r_k\| > \varepsilon_{threshold}$)~~

while ($\|r_k\| > \varepsilon_{threshold}$)

{

$$\alpha_k = -\frac{r_k^T P_k}{P_k^T H P_k} \quad \text{[Exact minimizer of } f \text{ along direction of } P_k\text{]}$$

$$x_{k+1} = x_k + \alpha_k P_k$$

$$r_{k+1} = Hx_{k+1} + c$$

$$\beta_{k+1} = \frac{P_k^T H r_{k+1}}{P_k^T H P_k}$$

$$P_{k+1} = -r_{k+1} + \beta_{k+1} P_k$$

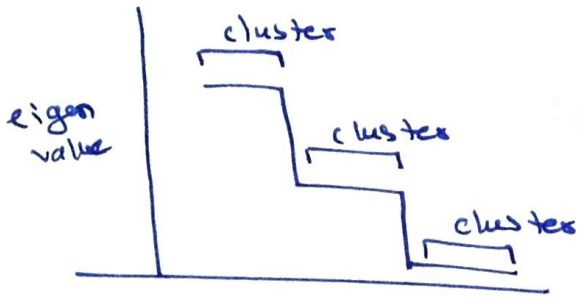
$$k = k+1$$

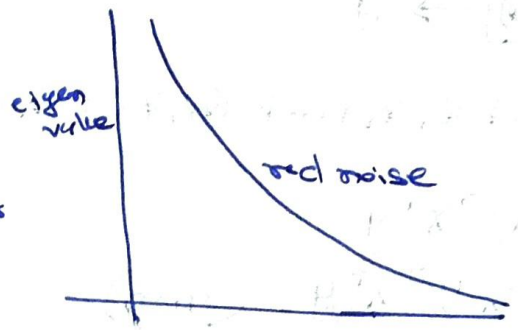
} end while.

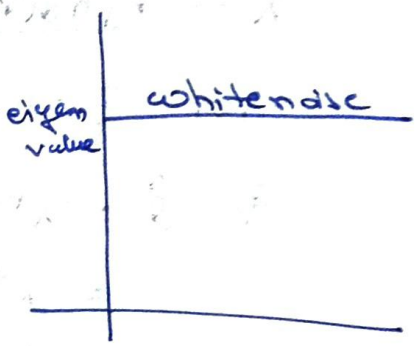
Remarks:

- For convex optimization problem, the above loop is run for atmost n times.
- This method has superlinear convergence. It is less expensive to perform (compared to Newton or quasi Newton). We never store the matrix H, but HP_k and Hx_k always.

Remarks: Conjugate gradient method works really well in case of matrices with clustered eigenvalues. (why?)







Pre-conditioned Conjugate Method (PCG):

why is it used?

$$Hx = -c$$

$$\Rightarrow M^{-1}Hx = -M^{-1}c$$

(where M is a pre-conditioner matrix chosen such that M is SPD and invertible)

We want $k(M^{-1}H) \ll k(H)$, k is condition number, or $M^{-1}H$ must have eigenvalue as clusters.

(If H is ill conditioned, we use $M^{-1}H$ instead of H, to make the problem a well-conditioned one.)

Constrained Optimization Problems

$X \longrightarrow \boxed{h_\theta(x;\theta)} \longrightarrow y$

let $\hat{y} = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n = X\theta$

$\Rightarrow \hat{\theta} = (X^T X)^{-1} X^T y$

$= X^{-1}(X^T)^{-1} X^T y \quad (m=n)$

$= X^{-1} y$

$\Rightarrow \hat{y} = X\theta = X X^{-1} y = y$

So in case of number of parameters = number of features (assuming 1 is also a feature), we may have zero error on the training data ($\|\hat{y} - y\|_2 = 0$).

let $m > n$. For example, suppose there are 1000 data points and 2 features (x_1, x_2). We create 998 new superficial features

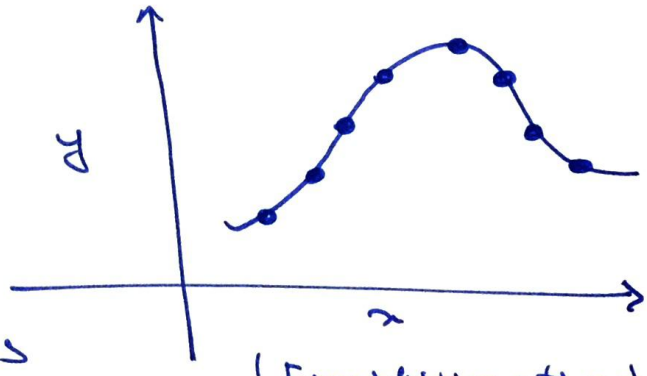
$f_j(x_1, x_2)$ – a function of x_1 and x_2 for each $j = 1, \dots, 998$.

Then we are in $m=n$ setting and can make the ~~test~~ training error zero. However, this is not good in practice as we have overfitted the data.

Read: Feature engineering, XGBoost.

If $m < n$, we can take some of the weights to be zero, in order to model the data.

- Put constraints on some parameters and make some of them zero, i.e. identify less relevant / redundant features



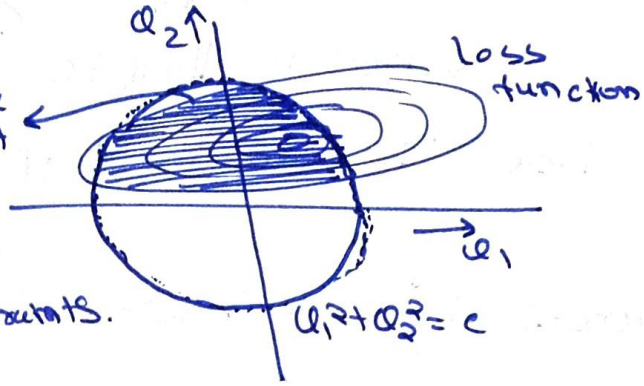
(Exact fitting the data)

- For $m < n$, we put constraints to prevent overfitting.

Our aim is to

$$\min_{\theta} \sum_{j=1}^{n} (\hat{y}_j - y_j)^2, \quad \text{where } \hat{y}_j = x_j^T \theta$$

constraints: $\|\theta\|_1 \le c_1$ or $\|\theta\|_2 \le c_2$ or $\|\theta\|_p \le c_p$.



Loss function = $\min_{\theta_1, \theta_2} L(\theta_1, \theta_2; D_{train})$ Feasible set

Feasible set - The set of values of θ which satisfy the constraints.

So, we need to

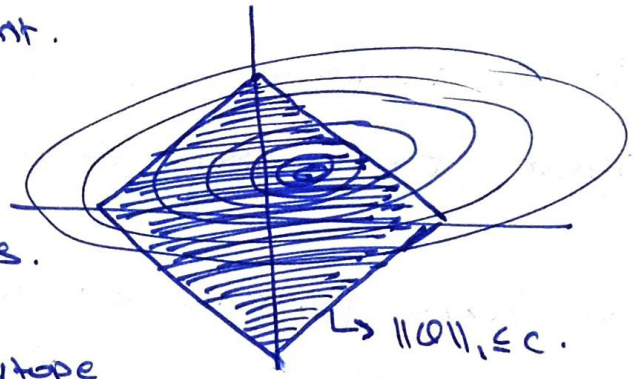
$$\min L(\theta; D_{train})$$

subject to

$$\|\theta\|_p \le c, \text{ for some } p \ge 1.$$

Let $p = 2$. Then we call it Ridge regression problem.

For $p = 1$, $\|\theta\|_1 \le c$ is the constraint.



Remarks: • $\|\cdot\|_1 \le c$ has only linear constraints, whereas

$\|\cdot\|_2 \le c$ has quadratic constraints.

- For $p = 1$, we have simplex polytope as feasible set and the minimiser is obtained at one of the vertex.
- Note that at any vertex, atleast one $\theta_j = 0$.
- $p = 1$ promotes sparsity in the parameter vector θ (i.e. a lot of components may be zero).
- This is called LASSO regression problem.
- LASSO, in general, is a good method of optimization.

Regularization:

- $L_{reg} = L + \alpha \|\theta\|_2$ (L_{reg} - regularized loss function)

- $\min L(\theta; D_{train})$
  subject to $\|\theta\|_1 \leq c$ $\iff \min \tilde{L} = L + \alpha(|\theta_1| + |\theta_2| - c)$

  Here $\tilde{L}$ is called the Lagrangian multiplier.

General Form of Constrained Optimization:

$\min f(x)$ subject to $c_i(x) = 0$, $i \in$ equality
$c_i(x) \geq 0$, $i \in$ inequality.

We denote the feasible set as Ω,

$$\Omega = \{x : c_i(x) = 0, \forall i \in \mathcal{E}_{eq}, c_i(x) \geq 0, \forall i \in \mathcal{E}_{ineq}\}.$$

- If $f(x)$ is linear and each $c_i(x)$ is linear, we call it a Linear Programming Problem.

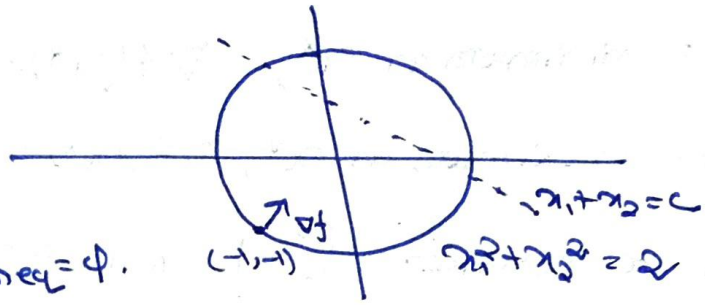
- If $f(x)$ is quadratic and each $c_i(x)$ is linear, we call it a Quadratic Programming Problem.

Read: Support vector machines, Active Set Method, LIBLINEAR, LIBSVC.

Active Set: $\mathcal{A} = \{i \in \mathcal{E} : c_i(x) = 0 \text{ holds}\}$.

Example: (Single Equality Constraint)

$\min (x_1 + x_2)$ s.t. $x_1^2 + x_2^2 = 2$.



minimizer = $(-1,-1)$.

Note that $\mathcal{E}_{eq} = \{1\}$, $\mathcal{E}_{ineq} = \phi$.

$\phi = \{1\}$.

let $f(x) = x_1 + x_2$.

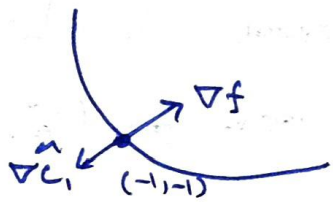
$\nabla f(x) = (1,1)$.

let $c_1(x) = x_1^2 + x_2^2 - 2$

$\Rightarrow \nabla c_1(x) = (2x_1, 2x_2)$

$\Rightarrow \nabla c_1(x)\big|_{(-1,-1)} = (-2,-2)$

$\Rightarrow \hat{\nabla} c_1(x)\big|_{(-1,-1)} = \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$.



Hence, at $(-1,-1)$, ∇f is parallel to ∇c_1 but in opposite direction.

At $(1,1)$, ∇f is parallel to ∇c_1 and in same direction.

In general we just need $\nabla f \parallel \nabla c$. (∵ we can take $-c$ instead of c)

$\Rightarrow \nabla f = \lambda \nabla c_1$, where $\lambda = \begin{cases} \pm 1, & \text{at point of minimum} \\ \pm 1, & \text{at point of maximum.} \end{cases}$

We take a small step s. Then $(x+s)$ is a point which

satisfies $c_1(x+s) = 0$.

$\Rightarrow c_1(x) + \nabla c_1(x)^T s = 0$ (first order Taylor)

$\Rightarrow \nabla c_1(x)^T s = 0 \quad \ldots (i)$

Since we want a decrease in f, $f(x+s) < f(x)$.

$\Rightarrow f(x) + \nabla f(x)^T s < f(x)$

$\Rightarrow \nabla f(x)^T s < 0 \quad \ldots (ii)$

(i) and (ii) leads to a contradiction. Hence $(-1,-1)$ is a point of local minimum.

$\Rightarrow$ For minimum at x^*, $\nabla f(x^*) = \lambda \nabla c_1(x^*)$ $(\lambda = \pm 1)$

For maximum at x^*, $\nabla f(x^*) = \lambda \nabla c_1(x^*)$ $(\lambda = \pm 1)$

Hence, we consider $\tilde{L} = L - \lambda c_1$ and find x^* such that

$\nabla \tilde{L}(x^*) = 0$ [Lagrangian].

We have now converted our constrained optimization problem to an unconstrained one.

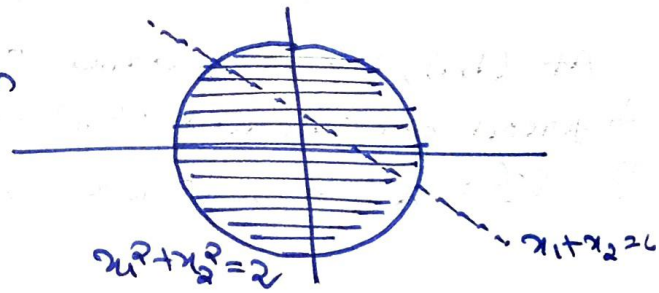
Remarks:

- $\frac{\partial \tilde{L}}{\partial \lambda} = 0 \Rightarrow c_1 = 0$ (Thus the constraint is satisfied)
- In general, $\tilde{L} = L - \sum_{j=1}^{m} \lambda_j c_j$ is the Lagrangian multiplier.

and we look for $(\lambda^*, x^*) = \arg\min_{(\lambda, x)} \tilde{L}(x, \lambda)$.

Example:

$\min x_1 + x_2$ subject to $x_1^2 + x_2^2 \leq 2$ (i.e. $2 - x_1^2 - x_2^2 \geq 0$)



As before, the point of minimum is $(-1,-1)$. Then,

$f(x+s) < f(x)$

$\Rightarrow f(x) + \nabla f(x)^T s < f(x)$

$\Rightarrow \nabla f(x)^T s < 0 \quad \ldots (i)$

Also, $c_1(x+s) \geq 0$

$\Rightarrow c_1(x) + \nabla c_1(x)^T s \geq 0$

$\Rightarrow \nabla c_1(x)^T s \geq -c_1(x) \quad (c_1(x) \geq 0)$

Let us consider the inequality

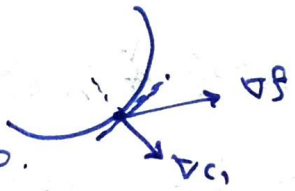
$\nabla c_1(x)^T s \geq 0$.

Case 1°: $\|x\| < 2$.

Then the point x is not an active point. If $\nabla f(x) \neq 0$, then choose $s = -\alpha \nabla f(x)$. This is a descent direction.

Case 2°: let $\|x\| = 2$.

We know, $c_1(x) + \nabla c_1(x)^T s \geq 0$.



s does not exist when $\nabla f = \lambda \nabla c, \lambda \geq 0$.

Let us now consider the Lagrangian

$\tilde{L} = f - \lambda c_1$ and impose extra condition that $\lambda c_1 = 0$.

$\Rightarrow \nabla \tilde{L} = \nabla f - \lambda \nabla c_1 = 0 \Rightarrow \nabla f = \lambda \nabla c_1$.

Remark°. If $c_1 \neq 0$, then $\lambda = 0$. $\Rightarrow \tilde{L} = f$ and it is enough to minimize f (like an internal point).

- If $c_1 = 0$, then λ may not be zero. (boundary point)

and we need the Lagrangian for obtaining minima.

- $\lambda = 0$, x is a degenerate point

Read°. Complementary Slackness

Note that till now we have only obtained an optimum point. To identify it as maximum/minimum, we need second order conditions.

Final Setting°:

$$L = f - \sum_{i=1}^{n} \lambda_i c_i,$$

$\Rightarrow \nabla L = 0$, $\lambda_i \geq 0$, when $i \in \mathcal{E}_{ineq}$

$\lambda_i c_i = 0, \forall i \in \mathcal{E}_{eq} \cup \mathcal{E}_{ineq}$

$c_i = 0, \forall i \in \mathcal{E}_{eq}$

$c_i \geq 0 \quad \forall i \in \mathcal{E}_{ineq}$

(*) This is called the first-order KKT conditions. We use this to solve a constrained optimization problem.

Remark: λ_i is also called **Sensitivity parameter.**

Example:

$\min x_1 + x_2^2$ such that $x_1^2 \leq 0$.

$L := x_1 + x_2^2 + \lambda x_1^2$, $\lambda \geq 0$, $\lambda x_1^2 = 0$.

$\nabla L = \frac{\partial L}{\partial x_1} = 1 + 2\lambda x_1$, $\frac{\partial L}{\partial x_2} = 2x_2$, $\frac{\partial L}{\partial \lambda} = x_1^2$

$\Rightarrow x_1 = x_2 = 0$. But $\lambda = -\frac{1}{2x_1}$ gives a problem.

This is because KKT is used when "constraint qualifications" is satisfied. Here $x_1^2 \leq 0$ is a redundant condition.

List of Constraint Qualifications:

- Linearity constraint qualification (LCQ)
- Linear independence constraint qualification (LICQ).
  - Gradients of active set constraints must be linearly independent.
-

Remarks If KKT conditions are satisfied, we ~~don't~~ need to check Second order conditions. The optimum point is ~~always~~ always a minimum.

- **Second order condition** – $\nabla^2 L$ is positive definite.

We must have $\omega^T \nabla^2 L \omega > 0$ whenever $\omega \in$ "critical cone".

- We use the second order condition in non linear problems.

Linear Programming

Consider a Nutrition problem

Eggs	Rice	Lentils	Veggies
c_1	c_2	c_3	c_4
x_1	x_2	x_3	x_4

c_j – ~~units~~ prices
x_j – ~~prices~~ units.

3 Nutritions: Proteins (b_1), Fats (b_2), Carbs (b_3).

	Protein	Fats	Carbs	Fibre
Eggs	a_{11}	a_{12}	a_{13}	a_{14}
Rice	a_{21}	a_{22}	a_{23}	a_{24}
Lentils	a_{31}	a_{32}	a_{33}	a_{34}
Veggies	a_{41}	a_{42}	a_{43}	a_{44}

~~Total cost~~ Aim is to ~~minimize~~ the expenditure. i.e.

$$\min_{x_1,x_2,x_3,x_4} (x_1c_1 + x_2c_2 + x_3c_3 + x_4c_4)$$

subject to $\sum_{i=1}^{4} a_{ij} x_i \geq b_j \quad ; j = 1,2,3.$

$x_i \geq 0 \quad \forall i = 1,2,3,4.$

Now suppose we add another constraint ~~$\sum a_{ij} x_j$~~ $\sum_{i=1}^{4} a_{i4} x_4 \geq b_4$.

The solution now gets changed. We aim to measure the sensitivity of the solution wrt this change.

Objective function: $\min_x c^T x$ subject to $A^T x \geq b$, $x \geq 0$.

- We have the inequality $A^T x \geq b$. We want to make it an equality by adding an error term. Such a variable is called slack variable.

$$\Rightarrow \min_x (c^T x + 0^T s) \quad \text{s.t. } A^T x - s = b,\ x \geq 0, s \geq 0$$

~~$\min_x A^T y$ s.t.~~

$$\Rightarrow \min_x D^T y \text{ s.t. } \tilde{A} y = b,\ y \geq 0 \quad \left(D = c^T \oplus 0^T,\ y = x \oplus s\right)$$

Remarks:

- If we are in a situation where the decision variables $x \in \mathbb{R}$

We decompose $x = x^+ - x^- = \max(x, 0) - \max(-x, 0)$.

- We could also substract/add a slack variable, depending on the situation.

Solving Linear Programming using KKT:

$\min c^T x$ s.t. $Ax = b$, $x \geq 0$ $\Rightarrow L(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x$.

$\Rightarrow \nabla L = c - A^T \lambda - s = 0 \Rightarrow A^T \lambda + s = c$.

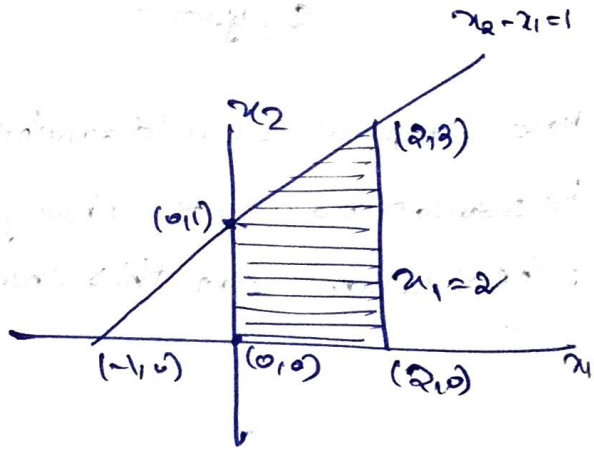
and $Ax = b$, $s \geq 0$, $x \geq 0$, $x_i s_i = 0$.

Simplex Algorithm:

Example: Consider $\min x_1 + x_2$

s.t. $x_2 - x_1 \leq 1$

$0 \leq x_1 \leq 2$, $x_2 \geq 0$



Adding Slack variables,

$$\min_{x_1,x_2,z_1,z_2} x_1 + x_2 + 0\cdot z_1 + 0\cdot z_2 = \min_{x} c^T x \qquad \left(c = (1,1,0,0)^t\right.$$
$$\left. x = (x_1, x_2, x_3 = z_1, x_4 = z_2)^t\right.$$

s.t. $x_2 - x_1 + z_1 = 1$

$x_1 + z_2 = 2$

$x_1 \geq 0,\ x_2 \geq 0,\ z_1 \geq 0,\ z_2 \geq 0.$

Then

$(0,0) \Rightarrow (0,0,1,2)$ $\quad [(x_1, x_2) \to (x_1, x_2, z_1, z_2)]$

$(2,0) \Rightarrow (2,0,3,0)$

$(2,3) \Rightarrow (2,3,0,0)$

$(0,1) \Rightarrow (0,1,0,2)$

$$\Rightarrow A = \left[\begin{array}{cc:cc} -1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 = z_1 \\ x_4 = z_2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Note that out the slack variables, the part of matrix A is identity $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Note: Since the minimum is obtained at one of the vertices, in simplex algorithm, we start at one random vertex and travel through all other vertices till the KKT conditions are satisfied at some vertex.

	x_1	x_2	x_3	x_4
Sol 1	0	1	0	2
Sol 2	2	3	0	0
Sol 3	2	0	3	0
Sol 4	0	0	1	2

} Note that no. of non zero entries in each point is equal to dimension of decision vector = 2.

- Basic Variables — x_1, x_2.
- Basis = $\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$
- Non-Basic Variables — z_1, z_2.

Algorithm:

- Start from a vertex
- Move from one vertex to another vertex
- Continue until RxT is satisfied.

Quadratic Optimization

Suppose there are n items in your portfolio and you decide to invest in x_i units of each. Let the respective returns be r_i (annually). Then the total return is

$$R = \sum_{i=1}^{n} x_i r_i$$

R is a random variable.

$$\Rightarrow E(R) = \sum x_i E(r_i).$$

Our aim is to maximize

$$\max E(R) - k \operatorname{var}(R) = \max \sum x_i E(r_i) - \lambda^T G x$$

$$\text{s.t.} \quad \sum x_i = 1, \; x_i \geq 0$$

(where $G = \sigma_i \sigma_j \rho_{ij}$)

$$\rho_{ij} = \frac{E[(r_i - \mu_i)(r_j - \mu_j)]}{\sigma_i \sigma_j}$$

$$\Rightarrow \max \mu^T x - k x^T G x,$$

$$\text{Subject to } \sum_{i=1}^{n} x_i = 1, \; x_i \geq 0$$

- This is a quadratic optimization problem.

Alternatively, $\max \mu^T x$ s.t. $x^T G x \leq c$ (bounding the variance)

or, $\min x^T G x$ s.t. $\mu^T x \geq t$

$$\sum x_i = 1$$

$$x_i \geq 0.$$

Formally, we have quadratic optimization problem as follows:

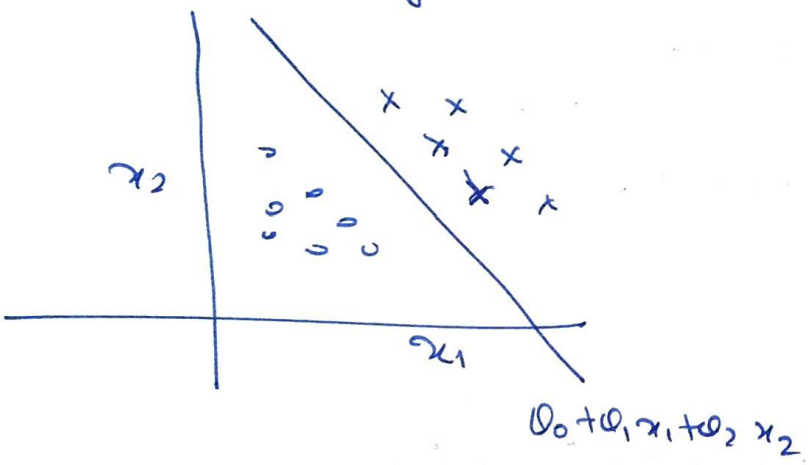
$$\min_{x \in \mathbb{R}^n} \quad q(x) = \frac{1}{2} x^T G x + c^T x$$

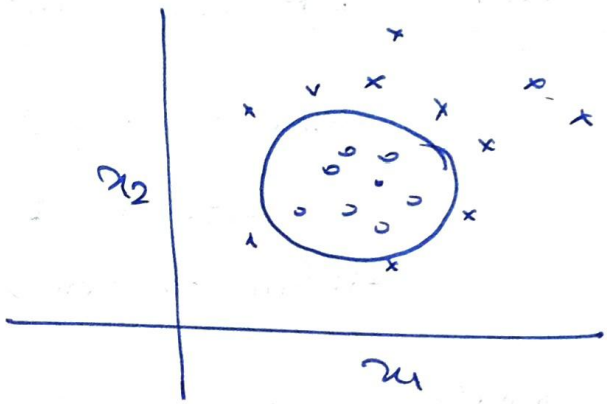
$$\text{s.t.} \quad a_i^T x = b_i \quad , i \in \mathcal{E}$$
$$a_i^T x \geq b_i \quad , i \in \mathcal{I}$$

- If G is symmetric and positive definite, we call it a quadratic optimization problem.

Support Vector Machines

It is used in classification problem by placing a decision boundary (linear or non linear)





Since there are multiple decision boundaries possible, we choose all such parallel boundaries and take their median.

Remarks For any data set in $\mathbb{R}^n$, there exists some $m > n$, such that the data set is separable in $\mathbb{R}^m$.

Reads Gaussian Kernel SVM, active set method, support vectors, VC dimension.

- Distance of datapoint $x^{(i)}$ from decision boundary $= \dfrac{t^{(i)} \cdot (w^T x^{(i)} + b)}{\|w\|}$, $t^{(i)} > 0$ if $x^{(i)}$ belongs to positive class else $t^{(i)} < 0$.

$$\frac{w^T x^{(i)} + b}{\|w\|} = \frac{w_1 x_1^{(i)} + w_2 x_2^{(i)} + b}{\sqrt{w_1^2 + w_2^2}}$$

~~⇒ use cvxopt~~

To find the decision boundary,

$$\max_j \; t^{(j)} \frac{w^T x^{(j)} + b}{\|w\|}.$$

We choose j such that $t^{(j)}[w^T x^{(j)} + b] = 1$ for some j.

$\Rightarrow$ $\max \frac{1}{\|w\|}$ subject to $t^{(j)}[w^T x^{(j)} + b] \geq 1$.

Remarks
- This is also a quadratic problem.
- Note that the constraints are all linear. The feasible set is a polytope. The solution lies on edge or vertex.

Equality Constrained Quadratic Problem

$$\min \; q(x) = \frac{1}{2} x^T G x + c^T x$$

$$\text{s.t. } Ax = b$$

(A is full row rank)

KKT: $L(x^*, \lambda^*) = \frac{1}{2} x^{*T} G x^* + c^T x^* - \lambda^{*T}(Ax^* - b)$

$$\nabla L = Gx^* + c - A^T \lambda^* = 0$$

$$Ax^* = b.$$

$$\Rightarrow \quad \begin{aligned} Gx^* - A^T \lambda^* &= -c \\ Ax^* &= b \end{aligned}$$

$$\Rightarrow \quad \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \quad \text{(KKT system)}$$

Notes
- $K = \begin{bmatrix} G & -A^* \\ A & 0 \end{bmatrix}$ is not singular matrix for a convex optimization problem.

Recall: Algorithm 16.3.

Simplex Method Algorithm

		c_1	c_2	c_3	c_4
		$E x_1$	$R x_2$	$L x_3$	$V x_4$
λ_1, b_1	P	a_{11}	a_{12}	a_{13}	a_{14}
λ_2, b_2	F	a_{21}	a_{22}	a_{23}	a_{24}
λ_3, b_3	C	a_{31}	a_{32}	a_{33}	a_{34}

Primal Problem:

$$\min_x \sum c_j x_j$$

Subject to $Ax \geq b$, $x \geq 0$.

Dual problem:

$$\max \text{Revenue} = \max_\lambda \sum \lambda_i b_i$$

Subject to $A^T \lambda \leq c$, $\lambda \geq 0$.

Remark:

- Primal and dual problems are equivalent. Both have the same KKT system.
- For any constrained optimization problem, we get a dual form. For a linear programming problem, the dual is unique.
- At the optimal point,

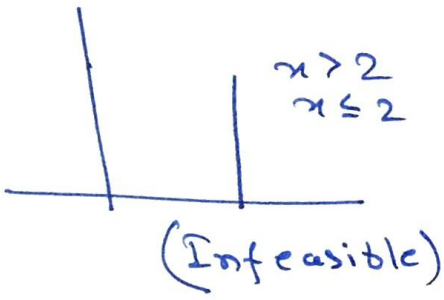
  value of primal = value of dual.

Duality Conditions:

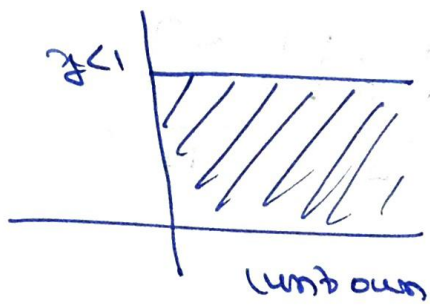
- If the primal has a solution, x^*, then the dual also has a solution λ^*, and

  $$c^T x^* = b^T \lambda^*.$$

- If the primal is unbounded, then the dual is infeasible.



(Infeasible)



(unbounded)

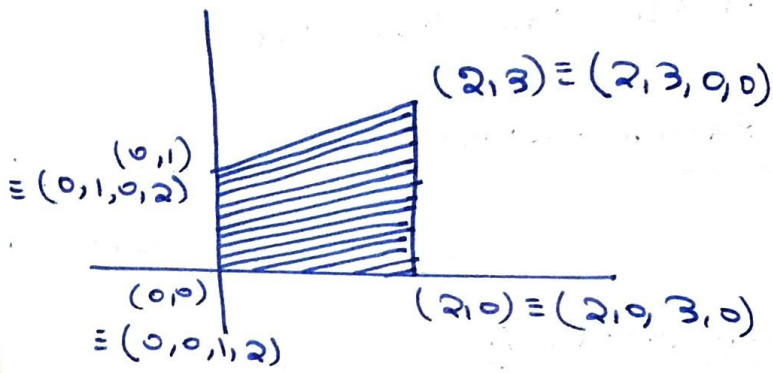
Example: $\min x_1 + x_2$

Subject to $x_2 - x_1 = 1$

$x_1 \leq 2$

$x_1, x_2 \geq 0$.

$$A = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



We split the solution (x_1, x_2, x_3, x_4) into x_B and x_N.

[B- Basic variables
N- Non-basic variables]

S = Slack variables for $Ax \geq b$.

S_B = Slack variables for basic

S_N = Slack variables for non-basic.

KKT: $\min c^T x$ s.t. $Ax = b, x \geq 0$.

$$L(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x.$$

- $A^T \lambda + s = c$
- $Ax = b$
- $x \geq 0$
- $s \geq 0$
- $x_i s_i = 0 \quad , \forall i$.

Note that $S_B = 0$, $S_N \neq 0$.

$$A = \begin{bmatrix} -1 & 1 & \vdots & 1 & 0 \\ 1 & 0 & \vdots & 0 & 1 \end{bmatrix}$$

B N

Revised Simplex Method

- Start from any vertex
- Move from vertex to vertex
- Iterate until all KKT conditions are satisfied.

Now,

$$Ax = b$$

$$\Rightarrow B x_B + N x_N = b$$

Also,

$$A^T \lambda + s = c$$

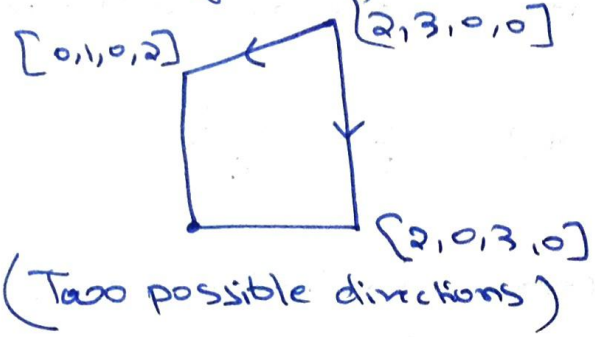
$$\Rightarrow B^T \lambda + s_B = c_B \quad \text{and} \quad N^T \lambda + s_N = c_N$$

$$\Rightarrow \lambda = (B^T)^{-1} c_B \quad (\because s_B = 0)$$

$$\Rightarrow s_N = c_N - (B^{-1} N)^T c_B .$$

Note that all conditions of KKT are satisfied, except possibly $s \geq 0$. If we have $s_N \geq 0$, then we are done.

Suppose $s_q < 0$ for some $q \in N$. Then we move q to basic set B and replace some index $p \in B$. This reduces the value of the cost function.



(Two possible directions)

⊛ Along the line ↑, value of q ↑ and value of x_2 ↓, until we reach another vertex.

Let x^+ be the new iterate. Then $Ax = Ax^+ = b$.

and $x_i^+ = 0, \forall i \in N \setminus \{q\}$.

$$Ax^+ = B x_B^+ + A_q x_q^+ = B x_B = Ax$$

$$\Rightarrow x_B^+ = x_B - B^{-1} A_q x_q^+ .$$

Also,

$$c^T x^+ = c_B^T x_B^+ + c_q x_q^+$$

$$= c_B^T x_B^* - c_B^T B^{-1} A_q x_q^+ + c_q x_q^+$$

$$= c_B^T x_B^* + [c_q - c_B^T B^{-1} A_q] x_q^+$$

$$= c_B^T x_B + s_q x_q^T \leq c_B^T x_B \quad [\because x_q \geq 0, s_q < 0]$$

Given B, N

$x_B = B^{-1} b$, $x_N = 0$

Stop if $B^{-1} b < 0$.

Solve for λ: $B^T \lambda = c_B$

Pricing s_N:

if $s_N \geq 0$:

stop (found solution)

else

q, $s_q < 0$ (choose q with $\max_q |s_q|$)

Solve $Bd = A_q$

if $d \leq 0$

stop (problem is unbounded)

else

$$x_q^+ = \min_i \frac{(x_B)_i}{d_i}$$

Remarks: How do we choose the initial point? Choosing it is itself an optimization problem — Two faced approach.

DEGENERACY:

If $x_B = B^{-1}b = 0$, it is called a degenerate problem. We may not be able to exit and an infinite loop occurs.

Quadratic Programming (Active Set Method):

$$\min_x \frac{1}{2}[x^T G x] + c^T x$$

$$\text{s.t.} \quad a_i^T x = b_i \quad , i \in \mathcal{E}$$
$$a_i^T x \geq b_i \quad , i \in \mathcal{I}.$$

$$\Rightarrow L(x, \lambda) = \frac{1}{2} x^T G x + c^T x - \sum_{i \in \mathcal{I} \cup \mathcal{E}} d_i (a_i^T x - b_i)$$

KKT conditions:

$$\nabla L = 0 \Rightarrow Gx^* + c - \sum_{i \in A(x^*)} d_i a_i = 0$$

$$a_i^T x^* = b_i \quad , i \in A(x^*)$$
$$a_i^T x_i > b_i \quad , i \in \mathcal{I} \setminus A(x^*)$$
$$d_i^* \geq 0 \quad , i \in \mathcal{I} \cap A(x^*).$$

$$\Rightarrow \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} \quad \end{bmatrix}$$

Example:

$$\min \frac{1}{2}[(x_1 - 3)^2 + (x_2 - 2)^2]$$

$$\text{s.t.} \quad -x_1 + x_2 \leq 0$$
$$x_1 + x_2 \leq 1.$$
$$-x_2 \leq 0$$

$$\frac{1}{2}[(x_1 - 3)^2 + (x_2 - 2)^2] = \frac{1}{2}x_1^2 + \frac{9}{2} - 3x_1 + \frac{x_2^2}{2} + \frac{4}{2} - \frac{4}{2}x_2$$

$$= \frac{1}{2} x^T G x + c^T x + b$$

$$= \frac{1}{2} x^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} -3 & -2 \end{bmatrix} x + \frac{13}{2}$$

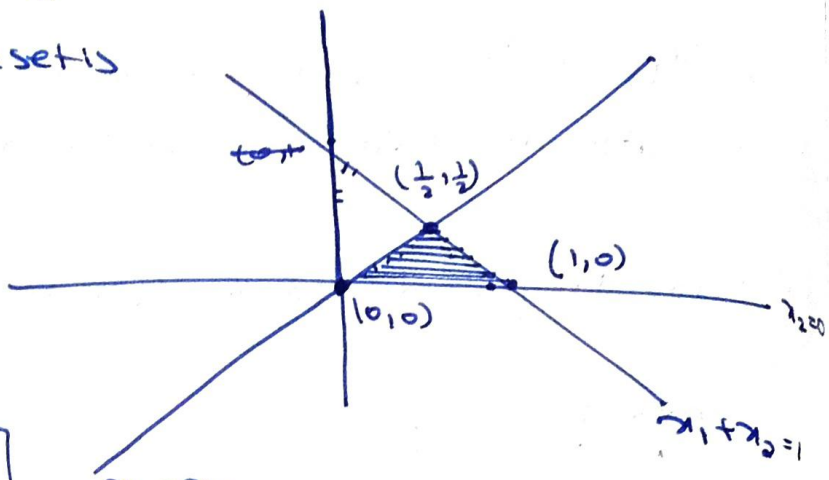
Let $x_0 = (0,0)$ be the initial guess.

For this point, the active set is

$\mathcal{A}(x_0) = \{1, 3\}$.

Then,

$$A = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}$$



$$\Rightarrow \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} Gx_0 + c \\ Ax_0 - b \end{bmatrix} = \begin{bmatrix} 0 + (-3) \\ 0 + (-2) \\ 0 + 0 \\ 0 + 0 \end{bmatrix} = \begin{bmatrix} -3 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \\ 5 \end{bmatrix} \Rightarrow p = (0,0), \quad \lambda^* = (3,5).$$

Step 2:

$W = \{1\}$ (ignoring $\max|\lambda_i|$)

$$x = [0,0], \quad A = [-1, 1]$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -p_1 \\ -p_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} -3 \\ -2 \\ 0 \end{bmatrix}$$

$$\Rightarrow \quad p_k = \left[\frac{5}{2}, \frac{5}{2}\right]$$

$$\Rightarrow x_{k+1} = x_k + \alpha p_k$$

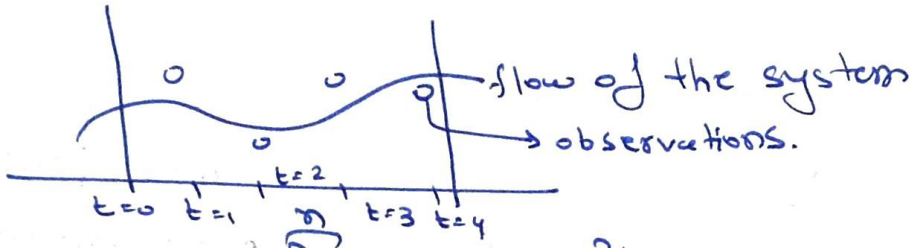
NVM:

$$\min \frac{1}{2}x_1^2 + x_2^2 - 3x_1 - 4x_2$$

$$\text{s.t.} \quad -2x_1 + x_2 \leq 0$$
$$x_1 + x_2 \leq 4$$
$$x_2 \geq 0$$

Ans: $x^* = \left(\frac{7}{3}, \frac{5}{3}\right)$. → Soln. not on vertex.

Derivative-Free Methods



$$\Psi(x_0) = \|x_0 - x_0^+\|_{B^{-1}} + \sum_{i=1}^{n} \|H_i(x_i) - y_i\|^2_{R^{-1}}, \quad x_i = M_{i-1}(x_{i-1}).$$

$H \rightarrow$ map from observed space to entire domain

$x_0 \rightarrow$ Initial point.

$x_{0b}^+ \rightarrow$ Previous knowledge of x_0.

Consider the Lagrangian multiplier,

$$L(x_0, \lambda) = \Psi(x_0) + \sum_{i=1}^{n} \lambda_i \left(x_i - M_{i-1}(x_{i-1})\right)$$

$$\nabla\Psi(x_0) = B^{-1}x_0 + \nabla f(x_0), \quad \text{where } f(x_0) = \sum \|H_i(x_i) - y_i\|^2_{R^{-1}}.$$

- Note that:

$$\frac{\partial H_i(x_i)}{\partial x_0} = \frac{\partial H_i(x_i)}{\partial x_i} \cdot \frac{\partial x_i}{\partial x_0}$$

$$= \frac{\partial H_i(x_i)}{\partial x_i} \cdot \frac{\partial}{\partial x_0} M(x_{i-1})$$

$$= \frac{\partial H_i(x_i)}{\partial x_i} \frac{\partial M}{\partial x_{i-1}} \cdot \frac{\partial x_{i-1}}{\partial x_0}$$

$$= \dots$$

Thus we can see that this technique is very costly. This inspires us to look for Derivative Free methods.

Why use Derivative Free methods?

(i) Calculating derivatives may ~~not~~ be expensive, as seen before.

(ii) Calculating derivatives may not be possible: For example, in LASSO regression. $L(x, \lambda) = f(x) + \lambda \sum |x_j|$

(iii) Hyperparameter tuning – We need to find the best hyperparameter value which minimizes functional value. But there is no particular relation between hyperparameters and objective function. We do a grid search algorithm in such case.

Derivative free Methods

Local Search

- Generalized pattern search

Global optimization

- Evolutionary algorithms
- Genetic algorithm.

Evolutionary Algorithms

It involves three steps –

(i) Generating population
(ii) Recombination
(iii) Neco population for next iteration.

Genetic Algorithm

It involves

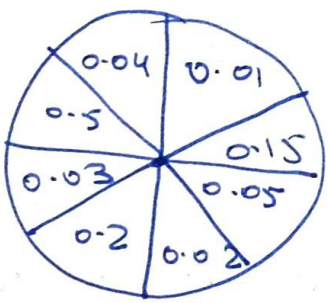
(i) Generate Population
(ii) Evaluate fitness score.
(iii) Selection (first do bit encoding)
(iv) Cross over
(v) Mutation
(vi) Neco generation.

Suppose we have 8 candidate solutions of $f(x_1, x_2, x_3)$.

	x_1	x_2	x_3
x_1	1	2	3
⋮	⋮		
x_8	0	1	0

$\rightarrow f_{\text{score}} = \dfrac{f_{\text{val}}}{\sum_{P_3} f_{\text{val}}}$

Roulette Selection Method



Arrange in increasing order. (least)

Choose x_j with cumulative $f_{\text{score}} > 0.6$

0.01
0.02
0.03
0.04
0.05
0.15 → 0.65
0.5 → 0.8

* Roulette Selection is highly biased towards higher f_{score}.

Tournament Selection

Cross Overs

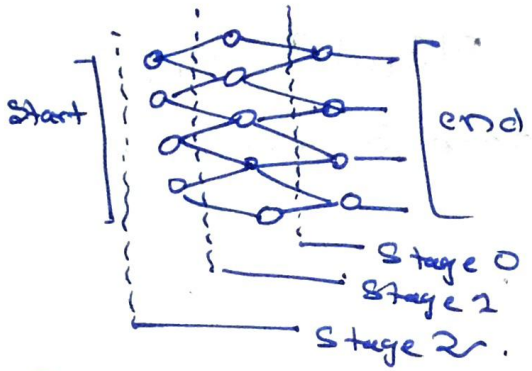
- One point, two points, N points uniform crossovers

Mutation

- We get 8 options after each ~~generate population~~ selection, crossover and mutation. Total = 24.

Convergence criteria — (i) Number of iteration
(ii) Number of iterations without improvement
(iii) Tolerance on objective function.

Dynamic Programming



- Each stage j has 4 states.

Formulation:

$$V_n(S_n) = \min \left[t_n(S_n) + V_{n-1}(S_{n-1}) \right]$$

Subject to $S_{n-1} = \begin{cases} S_n + 1, & \text{if we choose up and } n \text{ even} \\ S_n - 1, & \text{if we choose down and } n \text{ odd} \\ S_n, & \text{otherwise.} \end{cases}$

Markov Decision Process

$S = \{1, \ldots, 9\}$

$A = \{u, l, r, d\}$.

$P(2|S,u) = 1$ | $P(2|S,u) = 0.5$

$P(1|S,u) = 0$ | $P(3|S,u) = 0.5$

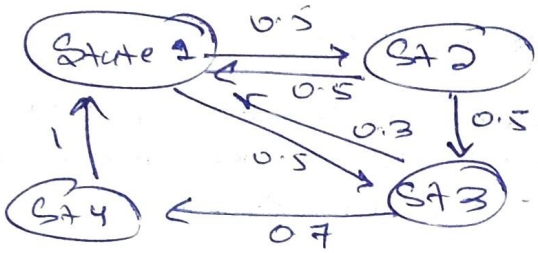
1	2	3
4	5	6
7	8	9

Reinforcement Learning (David Silverman)

- A process is Markov if $P(S_{t+1}|S_t) = P(S_{t+1}|S_1 \ldots S_t)$
- For a Markov State s and successor state s', the state transition probability is defined by

$$P_{ss'} = P(S_{t+1} = s' | S_t = s).$$

Transition matrix



$M =$

	1	2	3	4
1		0.5	0.5	0
2	0.5		0.5	0
3	0.3	0		0.7
4	1	0	0	

Markov Reward Process

Markov Reward process is a tuple (S, P, R, γ), where

(i) S is a finite set of states
(ii) P is state transition probability matrix } Markov process

$P_{ss'} = P(S_{t+1} = s' | S_t = s)$

(iii) R is reward function, $R_s = E(R_{t+1} | S_t = s)$
(iv) γ is a discount factor, $\gamma \in [0, 1]$.

Remarks We want to maximize cumulative reward in Reinforcement learning.

Definition: The return G_t is the total discounted reward from time step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k \geq 0} \gamma^k R_{t+k+1}$$

- The discount $\gamma \in [0,1]$ is the present value of future rewards.
- γ puts weight on how much we care to include the future rewards. $\gamma \approx 0 \Rightarrow G_t \approx R_{t+1}$.
  $\gamma \approx 1 \Rightarrow G_t \approx \sum_{k \geq 0} R_{t+k+1}$

Definition: The state value function $v(s)$ is

$$v(s) = E(G_t \mid S = s).$$

Bellman Equation

$$v(s) = E(G(t) \mid S = s) = E(R_{t+1} \mid S_t)$$

$$= E(R_{t+1} + \gamma v(S_{t+1}) \mid S = s).$$

$$= R_s + \gamma \sum_{s' \in S} P_{ss'} v(s')$$

$$\Rightarrow \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}.$$

$$\Rightarrow v \cdot (I - \gamma P) v = R$$

$$\Rightarrow v = (I - \gamma P)^{-1} R.$$

Markov Decision Process

Definition: Markov Decision Process is a tuple $\langle S, A, P, R, \gamma \rangle$

such that

(i) S is a finite set of states

(ii) A is a finite set of actions

(iii) P is a transition probability matrix

(iv) $P^a_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$

(v) R is a reward function

$$R^a_s = \mathbb{E}(R_{t+1} \mid S_t = s, A_t = a)$$

(vi) γ is a discounted factor $\gamma \in [0, 1]$.

Definition: (Formalizing what it means to take a decision)

A poly π is a distribution over actions given states

$$\pi(a|s) = P(A_t = a \mid S_t = s).$$

Remark: Policy only depends on the present state and not history. It is stationary (independent of t).

$$A_t \sim \pi(\cdot \mid S_t), \forall\, t \geq 0.$$

- Markov Reward Process can be recovered from Markov Decision process.

Suppose $S_1, S_2, \ldots$ is a Markov process $\langle S, P^\pi \rangle$. and the state reward sequence $S_1, R_1, S_2, \ldots$ is a Markov reward process $\langle S, P^\pi, R^\pi, \gamma \rangle$.

$$P^\pi_{ss'} = \sum_{a \in A} \pi(a|s)\, P^a_{s|s}$$

$$R^\pi_s = \sum_{a \in A} \pi(a|s)\, R^a_s.$$

Definition: The state-value function $v_\pi(s)$ of a MDP is the expected return starting from state s and then following policy π.

$$v_\pi(s) = E_\pi[G_t | S_t = s].$$

The action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, and following policy π.

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a].$$

Bellman Expectation Equation

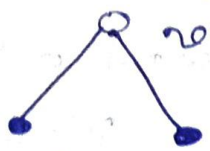
The state value function can be decomposed into immediate reward plus discounted value of successor state

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s].$$

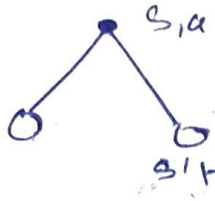
Similarly,
The action value function can be decomposed as

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, a_{t+1}) | S_t = s, a_t = a].$$

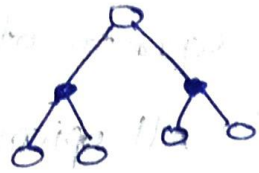
$$\Rightarrow v_\pi(s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$$



and $$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$



$$\Rightarrow v(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$



and $$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \left(\sum_{a' \in A} \pi(a'|s') q_\pi(s',a') \right)$$

Bellman Expectation Equation (Matrix form)

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

$$\Rightarrow v_\pi = (I - \gamma P^\pi)^{-1} R^\pi$$

Definition: (optimal value function)

The optimal state-value function $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a).$$

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem:

For any Markov Decision Process

(i) There exists an optimal policy π_* that is better than or equal to other policies, i.e. $\pi_* \geq \pi \ \forall \pi$.

(ii) All optimal policies ~~active~~ achieve the optimal value function $v_{\pi_*}(s) = v_*(s)$

(iii) All optimal policies achieve the optimal action-value function $q_{\pi_*}(s,a) = q_*(s,a)$.

Remark: An optimal policy can be found by maximising over $q_*(s,a)$.

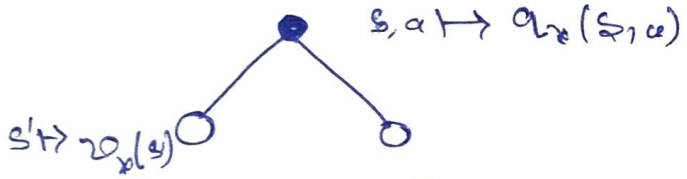
$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \underset{a \in A}{\operatorname{argmax}}\, q_*(s,a) \\ 0, & \text{otherwise.} \end{cases}$$

Bellman Optimality Equation

The optimal value functions are recursively related by the Bellman optimality equations:

$$v_*(s) = \max_a q_*(s,a)$$

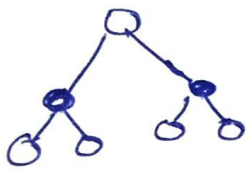
- Unlike Bellman expectation equation, where we look at the average of $q_*(s|a)$, here we take the maximum.

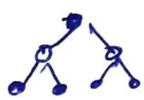


$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

Here we take the average, and not maximum.

$\Rightarrow$





~~$v_*(s) = \max$~~ $q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s',a')$

$$v_*(s) = \max_a \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s') \right]$$