Machine Learning

Unit 5

* Principal Component Analysis

→ PCA is a statistical method used to simplify data

→ It reduces the dimensionality of the data while retaining most of the variation.

→ used for dimensionality reduction, data visualization, noise filtering and feature extraction.

Principal Components

→ New variables are constructed as linear combinations / mixtures of the initial variables.

→ These combinations are done in such a way that the principal components are uncorrelated.

→ Most of the information within the initial variables is squeezed into the first components.

→ Principal components represent the direction of the data that explain a maximal amount of variance.

First principal component = largest variance

Second principal component = $\perp^r$ to first principal component — has the next highest variance.

Key Aspects of PCA

→ reduce dimensionality without ~~reducing~~ losing much info

→ This is achieved by discarding the components with low information and considering the remaining components as new variables

→ The principal components are less interpretable & don't have any real meaning.

* Steps in PCA

(1) Standardization

→ PCA is sensitive regarding the variances of initial variance.

eg. variables that range between 0- 100 dominate those between 0 and 1

→ Transform data to comparable scales.

(2) Covariance Matrix Computation

→ identify relationships between variables → sometimes variables are highly correlated & have redundant information

→ cov matrix is a d x d symmetric matrix

$$\begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

Step (3) Computation of eigen values and eigen vectors

→ The eigen vectors of the covariance matrix are the directions of the axes where there is the most variance (most info)

→ The eigen values are the coefficients attached to the eigenvectors, which give the amount of variance in each principal component

→ Rank the eigen vectors in order of their eigenvalues to get PCs in order of significance.

(4) Create a feature vector

→ Column matrix of eigen vectors that one chooses to keep

→ choose only p eigenvectors

(5) Recast data along principal components

→ Reorient data from the original axes to the ones represented by the principal components.

$$\text{FinalDataset} = \text{FeatureVector}^{T} * \text{Standardized OG Dataset}$$

*PCA @hallenges

1. Interpretability
2. Assumptions - assumes linear relationships between variables and normally distributed data.
3. Overfitting - can overfit data if too many principal components are retained.

* Linear Discriminant Analysis

→ dimensionality reduction and classification technique

PCA => maximize variance

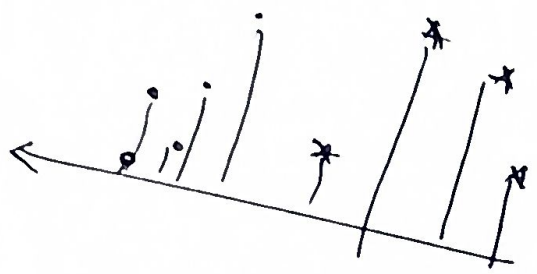
LDA => supervised algorithm which uses both the input data and labels to maximize the separation between multiple classes.

LDA Aim : project data onto a lower dimensional space while preserving class separability

maximize the ratio of between class scatter to within-class scatter

* Two class LDA Problem

Given a training dataset $x_1, x_2 \ldots x_n \in R^d$ consisting of 2 classes C_1, C_2. find a unit vector direction that best discriminates between the 2 classes



* LDA Steps

1. Compute class means
2. Compute the within class scatter matrix (S_w)
3. between class scatter matrix (S_b)
4. Compute the eigen values & eigen vectors of $S_w^{-1}S_b$. Those are the directions that maximize class separation
5. Select linear discriminants - choose top k eigen vectors
6. Project data onto new feature subspace.

Advantages

↓ dimensionality, preserve class discriminability

can handle multicollinearity

simple to implement

useful for both classification & dimensionality reduction

LDA Numericals

1. Compute the linear Discriminant projection for the following 2D dataset

$X_1 = \{(4,2), (2,4), (2,3), (3,6), (4,4)\}$

$X_2 = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$

Step 1 : **Find class means**

$$\mu_1 = \left\{\begin{pmatrix}4\\2\end{pmatrix} + \begin{pmatrix}2\\4\end{pmatrix} + \begin{pmatrix}2\\3\end{pmatrix} + \begin{pmatrix}3\\6\end{pmatrix} + \begin{pmatrix}4\\4\end{pmatrix}\right\} / 5$$

$$= \begin{pmatrix}3\\3.8\end{pmatrix}$$

$$\mu_2 = \left\{\begin{pmatrix}9\\10\end{pmatrix} + \begin{pmatrix}6\\8\end{pmatrix} + \begin{pmatrix}9\\5\end{pmatrix} + \begin{pmatrix}8\\7\end{pmatrix} + \begin{pmatrix}10\\8\end{pmatrix}\right\} / 5$$

$$= \begin{pmatrix}8.4\\7.6\end{pmatrix}$$

Step 2 : **Compute the covariance matrix for both classes**

$$S_1 = \sum_{x \in \omega_1} (x - \mu_1)(x - \mu_1)^T$$

$$= \left[\begin{pmatrix}4\\2\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^2 + \left[\begin{pmatrix}2\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^2 + \left[\begin{pmatrix}2\\3\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^2 +$$

$$\left[\begin{pmatrix}3\\6\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^2 + \left[\begin{pmatrix}4\\4\end{pmatrix} - \begin{pmatrix}3\\3.8\end{pmatrix}\right]^2$$

$$\begin{pmatrix}1\\-1.8\end{pmatrix}\begin{pmatrix}1 & -1.8\end{pmatrix} + \begin{pmatrix}-1\\0.2\end{pmatrix}\begin{pmatrix}-1 & 0.2\end{pmatrix} + \begin{pmatrix}-1\\-0.8\end{pmatrix}\begin{pmatrix}-1 & -0.8\end{pmatrix}$$

$$\begin{pmatrix}0\\2.2\end{pmatrix}\begin{pmatrix}0 & 2.2\end{pmatrix} + \begin{pmatrix}1\\0.2\end{pmatrix}\begin{pmatrix}1 & 0.2\end{pmatrix}$$

$$= \begin{bmatrix}1 & -1.8\\ -1.8 & 3.24\end{bmatrix} + \begin{bmatrix}1 & -0.2\\ -0.2 & 0.04\end{bmatrix} + \begin{bmatrix}1 & 0.8\\ 0.8 & 0.64\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 4.84\end{bmatrix}$$

$$+ \begin{bmatrix}1 & 0.2\\ 0.2 & 0.04\end{bmatrix}$$

$$= \begin{bmatrix}4 & -1\\ -1 & 8.8\end{bmatrix}$$

$$\therefore \quad S_1 = \begin{bmatrix}4 & -1\\ -1 & 8.8\end{bmatrix}$$

$$S_2 = \sum_{x \in \omega_2} (x-\mu)(x-\mu)^T$$

$$= \left[\begin{pmatrix}9\\10\end{pmatrix} - \begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^2 + \left[\begin{pmatrix}6\\8\end{pmatrix} - \begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^2 + \left[\begin{pmatrix}9\\6\end{pmatrix} - \begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^2$$

$$+ \left[\begin{pmatrix}8\\7\end{pmatrix} - \begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^2 + \left[\begin{pmatrix}10\\8\end{pmatrix} - \begin{pmatrix}8.4\\7.6\end{pmatrix}\right]^2$$

$$= \begin{pmatrix} 0.6 \\ 2.4 \end{pmatrix} \begin{pmatrix} 0.6 & 2.4 \end{pmatrix} + \begin{pmatrix} -2.4 \\ 0.4 \end{pmatrix} \begin{pmatrix} -2.4 & 0.4 \end{pmatrix} + \begin{pmatrix} 0.6 \\ -2.6 \end{pmatrix} \begin{pmatrix} 0.6 & -2.6 \end{pmatrix}$$

$$+ \begin{pmatrix} -0.4 \\ -0.6 \end{pmatrix} \begin{pmatrix} -0.4 & -0.6 \end{pmatrix} + \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \begin{pmatrix} 0.6 & 0.4 \end{pmatrix}$$

$$= \begin{bmatrix} 0.36 & 1.44 \\ 1.44 & 5.76 \end{bmatrix} + \begin{bmatrix} 5.76 & -0.96 \\ -0.96 & 0.16 \end{bmatrix} + \begin{bmatrix} 0.36 & -1.56 \\ -1.56 & 6.76 \end{bmatrix}$$

$$+ \begin{bmatrix} 0.16 & 0.24 \\ 0.24 & 0.36 \end{bmatrix} + \begin{bmatrix} 0.36 & 0.24 \\ 0.24 & 0.16 \end{bmatrix} \begin{bmatrix} 2.56 & 0.64 \\ 0.64 & 0.16 \end{bmatrix}$$

$$= \begin{bmatrix} 9.2 & -0.2 \\ -0.2 & 13.2 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 9.2 & -0.2 \\ -0.2 & 13.2 \end{bmatrix}$$

Step 3 | Compute S_W

$$S_W = S_1 + S_2 = \begin{bmatrix} 4 & -1 \\ -1 & 8.8 \end{bmatrix} + \begin{bmatrix} 9.2 & -0.2 \\ -0.2 & 13.2 \end{bmatrix} = \begin{bmatrix} 13.2 & -1.2 \\ -1.2 & 22 \end{bmatrix}$$

Step 4 : Compute $S_w^{-1}(\mu_1 - \mu_2)$

$$S_w = \begin{bmatrix} 13.2 & -1.2 \\ -1.2 & 22 \end{bmatrix}$$

$$S_w^{-1} = \frac{1}{|S_w|} \text{adj}(S_w)$$

$$= \frac{1}{288.96} \begin{bmatrix} 22 & 1.2 \\ 1.2 & 13.2 \end{bmatrix}$$

$\mu_1 - \mu_2$

$$\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}$$

$$= \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix}$$

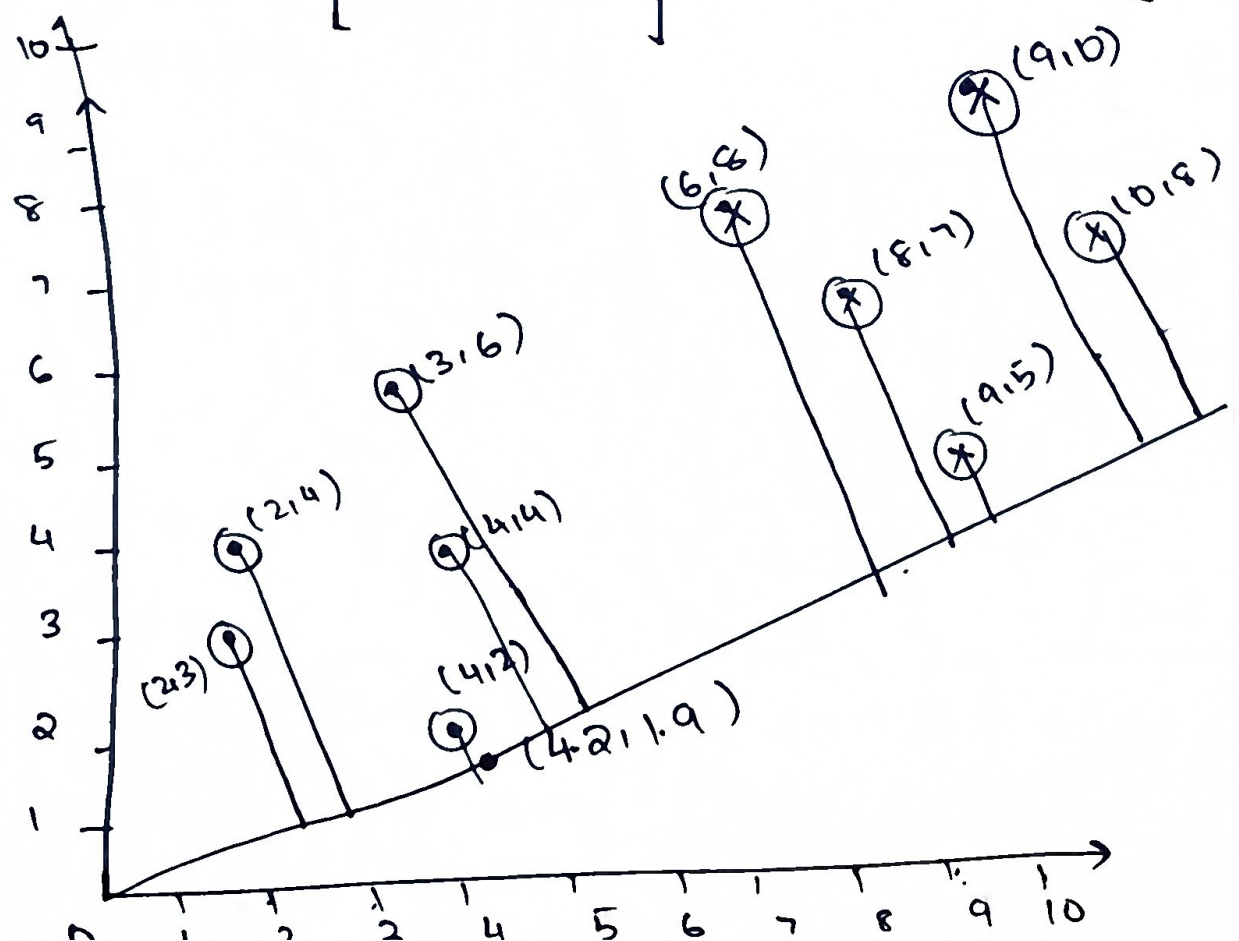
$S_w^{-1}(\mu_1 - \mu_2)$

$$= \frac{1}{288.96} \begin{bmatrix} 22 & 1.2 \\ 1.2 & 13.2 \end{bmatrix} \begin{bmatrix} -5.4 \\ -3.8 \end{bmatrix}$$

2x2 2x1

$$= \frac{1}{288.96} \begin{bmatrix} -123.36 \\ -56.64 \end{bmatrix} = \begin{bmatrix} 0.42 \\ 0.19 \end{bmatrix}$$

project as (4.2, 1.9)



Q2. CAT-2 Consider the given samples with 2 classes: ω_1 and ω_2. Apply LDA.

$$\omega_1 = \begin{pmatrix}3\\5\end{pmatrix}, \begin{pmatrix}4\\3\end{pmatrix}, \begin{pmatrix}5\\6\end{pmatrix}, \begin{pmatrix}7\\5\end{pmatrix}$$

$$\omega_2 = \begin{pmatrix}9\\4\end{pmatrix}, \begin{pmatrix}10\\1\end{pmatrix}, \begin{pmatrix}12\\3\end{pmatrix}, \begin{pmatrix}13\\6\end{pmatrix}$$

Step 1 : **Compute mean for each class**

$$\mu_1 = \left\{\begin{pmatrix}3\\5\end{pmatrix} + \begin{pmatrix}4\\3\end{pmatrix} + \begin{pmatrix}5\\6\end{pmatrix} + \begin{pmatrix}7\\5\end{pmatrix}\right\} / 4 = \begin{pmatrix}4.75\\4.75\end{pmatrix}$$

$$\mu_2 = \left\{\begin{pmatrix}9\\4\end{pmatrix} + \begin{pmatrix}10\\1\end{pmatrix} + \begin{pmatrix}12\\3\end{pmatrix} + \begin{pmatrix}13\\6\end{pmatrix}\right\} / 4 = \begin{pmatrix}11\\3.250\end{pmatrix}$$

Step 2 : **Compute the covariance matrices**

$$S_1 = \sum_{x_i \in \omega} (x_i - \mu)(x_i - \mu)^T$$

$$= \left[\begin{pmatrix}3\\5\end{pmatrix} - \begin{pmatrix}4.75\\4.75\end{pmatrix}\right]^2 + \left[\begin{pmatrix}4\\3\end{pmatrix} - \begin{pmatrix}4.75\\4.75\end{pmatrix}\right]^2 + \left[\begin{pmatrix}5\\6\end{pmatrix} - \begin{pmatrix}4.75\\4.75\end{pmatrix}\right]^2 + \left[\begin{pmatrix}7\\5\end{pmatrix} - \begin{pmatrix}4.75\\4.75\end{pmatrix}\right]^2$$

$$= \begin{pmatrix}-1.75\\0.25\end{pmatrix}\begin{pmatrix}-1.75 & 0.25\end{pmatrix} + \begin{pmatrix}-0.75\\-1.75\end{pmatrix}\begin{pmatrix}-0.75 & -1.75\end{pmatrix} + \begin{pmatrix}0.25\\1.25\end{pmatrix}\begin{pmatrix}0.25 & 1.25\end{pmatrix}$$

$$+ \begin{pmatrix}2.25\\0.25\end{pmatrix}\begin{pmatrix}2.25 & 0.25\end{pmatrix}$$

$$= \begin{bmatrix}3.0625 & 0.4375\\ -0.4375 & 0.0625\end{bmatrix} + \begin{bmatrix}\quad & \quad\\ \quad & \quad\end{bmatrix} + \begin{bmatrix}\quad & \quad\\ \quad & \quad\end{bmatrix} + \begin{bmatrix}\quad & \quad\\ \quad & \quad\end{bmatrix}$$

$$\begin{bmatrix} 3.0625 & -0.4375 \\ -0.4375 & 0.0625 \end{bmatrix} + \begin{bmatrix} 0.375 & 1.3125 \\ 0.875 & 3.0625 \end{bmatrix} + \begin{bmatrix} 0.0625 & 0.3125 \\ 0.3125 & 1.5625 \end{bmatrix}$$

$$+ \begin{bmatrix} 5.0625 & 0.5625 \\ 0.5625 & 0.0625 \end{bmatrix}$$

$$= \begin{bmatrix} 8.5625 & 10.75 \\ 1.3125 & 4.75 \end{bmatrix}$$

$$S_2 = \sum_{x_i \in \omega_2} (x_i - \mu)(x_i - \mu)^T$$

$$\left[\begin{pmatrix} 9 \\ 4 \end{pmatrix} - \begin{pmatrix} 11 \\ 3.5 \end{pmatrix}\right]^2 + \left[\begin{pmatrix} 10 \\ 1 \end{pmatrix} - \begin{pmatrix} 11 \\ 3.5 \end{pmatrix}\right]^2 + \left[\begin{pmatrix} 12 \\ 3 \end{pmatrix} - \begin{pmatrix} 11 \\ 3.5 \end{pmatrix}\right]^2 +$$

$$\left[\begin{pmatrix} 13 \\ 6 \end{pmatrix} - \begin{pmatrix} 11 \\ 3.5 \end{pmatrix}\right]^2$$

$$= \begin{pmatrix} -2 \\ 0.5 \end{pmatrix}\begin{pmatrix} -2 & 0.5 \end{pmatrix} + \begin{pmatrix} -1 \\ -2.5 \end{pmatrix}\begin{pmatrix} -1 & 2.5 \end{pmatrix} + \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}\begin{pmatrix} 1 & 0.5 \end{pmatrix}$$

$$+ \begin{pmatrix} 2 \\ 2.5 \end{pmatrix}\begin{pmatrix} 2 & 2.5 \end{pmatrix}$$

$$= \begin{bmatrix} 4 & -1 \\ -1 & 0.25 \end{bmatrix} + \begin{bmatrix} 1 & -2.5 \\ -2.5 & -6.25 \end{bmatrix} + \begin{bmatrix} 1 & 0.5 \\ -0.5 & -0.25 \end{bmatrix} +$$

$$\begin{bmatrix} 4 & 5 \\ 5 & 6.25 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & 2 \\ 1 & 0 \end{bmatrix}$$

$$S_W = \begin{bmatrix} 18.5625 & 3.75 \\ 2.3125 & 4.75 \end{bmatrix}$$

$$\mu_1 - \mu_2 = \begin{pmatrix} 4.75 \\ 4.75 \end{pmatrix} - \begin{pmatrix} 11 \\ 3.5 \end{pmatrix} = \begin{pmatrix} -6.25 \\ 1.25 \end{pmatrix}$$

$$S_W^{-1}(\mu - \mu_2) = \frac{1}{\quad} \begin{bmatrix} 4.75 & -3.75 & -2.3125 \\ 3.75 & 18.5625 \end{bmatrix} \begin{bmatrix} -6.25 \\ 1.25 \end{bmatrix}$$

* Gradient Descent Method

→ Gradient descent is a way to minimize an objective fn.

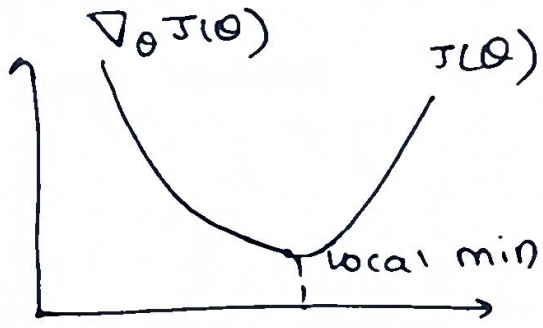
$J(\theta)$

$J(\theta)$ = objective fn

θ = model params $\in R^d$

η = learning rate – determines the no. of steps needed to reach a local minimum.

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$



Types of Gradient Descent → depends upon the amount of data in $\nabla_\theta J(\theta)$ - depending on the amt of data, there is a tradeoff between accuracy & time.

(i) Batch gradient descent

→ For each update, gradients of the whole dataset have to be found

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

Advantage : for convex error ⇒ converges to global min, for non-convex surfaces → converges to local min

Disadvantage: slow, does not fit in memory

(ii) Stochastic gradient descent

→ perform a parameter update for each training example $x^{(i)}$ & label $y^{(i)}$

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

(decreasing LR ⇒ convergence similar to BGD)

Advantage - much faster, can learn online

Disadvantage - high variance, objective fn fluctuates a lot

(iii) Mini-batch gradient descent

→ takes the best of both BGD & SGD, performs updates on each mini batch n

$$\theta = \theta - \eta \nabla_\theta J(\theta : x^{(i:i+n)}, y^{(i:i+n)}$$

Advantage : reduces variances, more stable convergence

Disadvantage : need to set mini batch size

* Fluctuations in Batch vs. SGD

BGD - small fluctuations

SGD - larger fluctuation, but enables it to find a potentially better new minima.

Gradient Descent Numerical

SAT Let $f(x) = x^2 - 4x + 3$. Find the gradient descent function value for 3 steps. Let $x_0 = 3$ and $\eta = 0.1$

$f(x) = x^2 - 4x + 3$

$f'(x) = 2x - 4$

$F'(x_0) = 2(3) - 4 = \underline{2}$

$x_{old} = 3$

$f'(x_{old}) = 2$

(i) $x_{new} = x_{old} - \eta F'(x_{old})$

$= 3 - (0.1)(2)$

$= 3 - 0.2$

$= \underline{2.8}$

(ii) $x_1 = 2.8$

$f'(x_1) = 2(2.8) - 4$

$= 5.6 - 4$

$= 1.6$

$x_{new} = 2.8 - (0.1)(1.6)$

$= 2.8 - 0.16$

$= \underline{2.64}$

(iii) $x_3 = 2.64$

$f'(x_3) = 5.28 - 4 = 1.28$

$x_3\ x_{new} = 2.64 - (0.1)(1.28)$

$= \underline{2.512}$

$\therefore x_1 = 2.8$

$x_2 = 2.64$

$x_3 = 2.512$

* Conjugate Gradient Descent

1. Given $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

$X_1 = (0,0)$

use the conjugate gradient method for 3 iterations to find the descent

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix}$$

$$\nabla f = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{Bmatrix}$$

$X_1 = (0,0)$ sub X_1

$$\therefore \nabla f_1 = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

$$S_1 = -\nabla f_1 = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}$$

Compute the Hessian matrix

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

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

Compute the step length $\lambda_1 = \dfrac{S_1^T S_1}{S_1^T H S_1}$

$$\lambda_1 = \frac{\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}}{\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}}$$

$$= \frac{2}{\begin{bmatrix} -2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}} = \frac{[2]}{[2]} = 1$$

compute x_{i+1}

$$x_2 = x_1 + \lambda_1 s_1$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\boxed{x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}}$$

optimality check

$$\nabla f(x_2) = \begin{Bmatrix} -1 \\ -1 \end{Bmatrix} \neq 0 \Rightarrow \text{not converged}$$

Iteration 2

$$\nabla f_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$s_2 = -\nabla f_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = \frac{s_2^T s_2}{s_2^T H s_2}$$

$$\lambda_2 = \frac{\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}}{\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \frac{2}{\begin{bmatrix} 6 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \frac{2}{10} = \underline{0.2}$$

$$x_3 = x_2 + \lambda_2 s_2$$

$$= \begin{bmatrix} -1 \\ -1 \end{bmatrix} + 0.2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\boxed{x_3 = \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix}}$$

continue

* Hill climbing

→ most basic local search technique.

→ At each step, the current node is replaced with the best value.

If the heuristic h is:

(i) cost ⇒ look for local minima

(ii) value ⇒ look for local maxima

ALGO

Function HILL-CLIMBING (problem) ~~return~~ returns a state that is a local maximum

current ← MAKENODE (problem.INITIAL STATE)

loop do

neighbor ← highest valued successor of current

if neighbor.VALUE ≤ current.VALUE return current.STATE

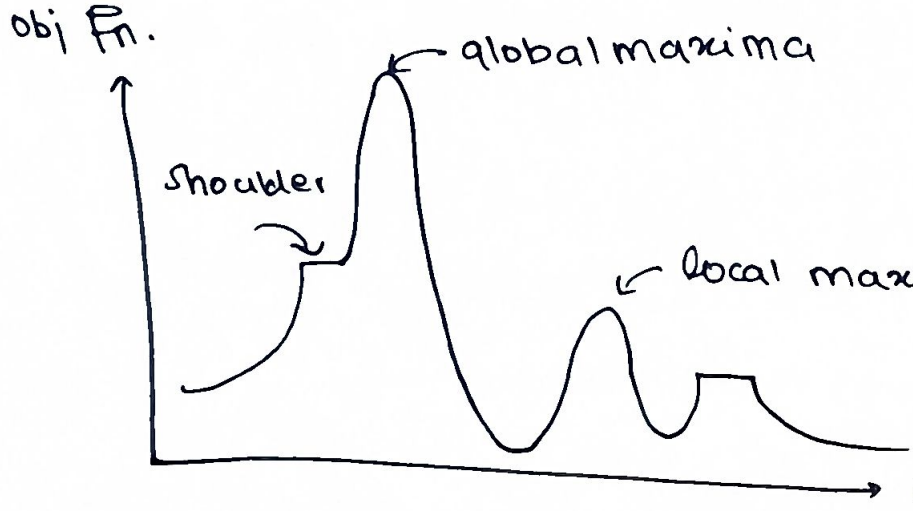
else current ← neighbor

→ Hill climbing is also called <u>greedy local search</u>

→ It may get stuck due to : (i) local maxima

(ii) ridges (sequence of local maxima)

(iii) plateaus



Variants (i) Stochastic hill climbing

(ii) First choice hill climbing

(iii) Random restart hill climbing

* Simulated Annealing

→ a version of stochastic hill climbing, where some downhill moves are allowed.

→ Based on the concept of cooling of metals at a controlled rate. The slowly falling temperature allows the atoms to line themselves up and form a regular crystalline structure. However, if the temperature goes down too quickly, the atoms do not have time to reorient themselves in a regular structure and the structure is amorphous w/ higher energy.

→ SA allows function value fluctuations that lead to a point w/ higher energies early on.

→ At lower temperatures, SA evaluates only at local points.

→ SA relies most on the annealing schedule, which specifies how rapidly the temperature is cooled