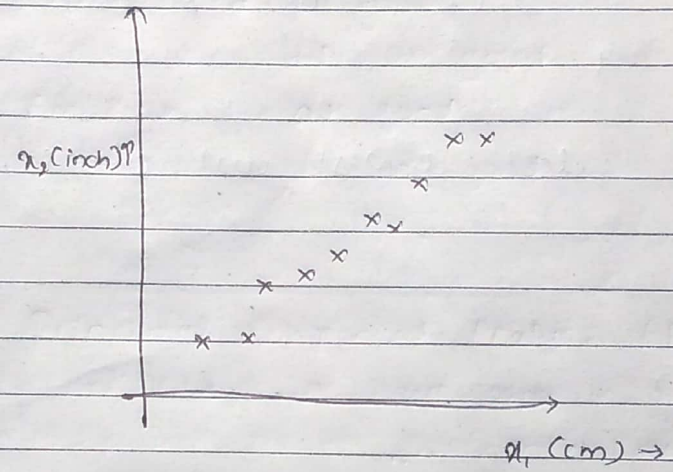
Dimensionality reduction

Dimensionality reduction helps us not only to reduce the memory usage but also the learning speed is improved.

Let's say there are 2 features length in cms & length in inches of same object. both rounded to nearest value.

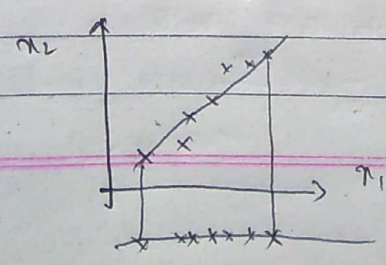
We can reduce this to one feature

In a bigger problem we have loads of features hence it is difficult to keep track



for example lets say in plane driving example : if Pilot's skill, Pilot's enjoyment are the features they are highly correlated and can be clubbed together to form a new feature 'aptitude'.

So in above example we draw a st. line that fits the curve & drop the corresponding pts on new feature



Thus each feature will be indicated just by one number rather than 2 hence we ~~halfed~~ reduce memory requirement to half.

Similarly data reduction can be from 3D to 2D. or in other words we reduce k dimensional data to c dimensional (c<k).

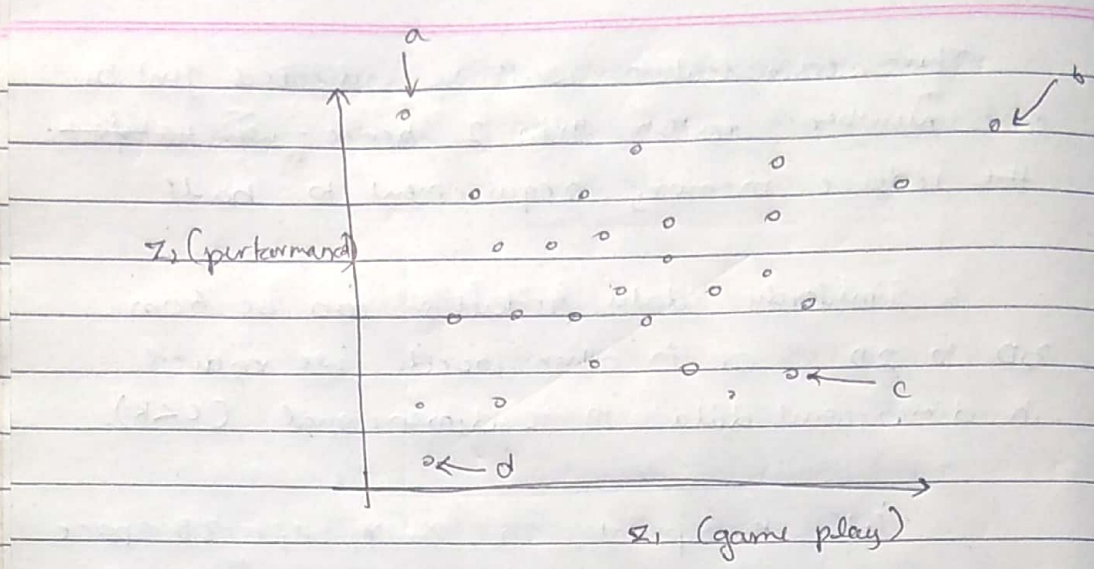
Let's the points in x_1, x_2, x_3 3D space lie on a single plane then this can be projected on a plane z_1, z_2.

Data visualization

Let's say we have huge no. of features & huge number of dataset. This can not be plotted let's say there are 50 features, we reduce it to which summerizes the data. Thus we can plot the data & visualize the trend.

Let's take an example of a TT player Mathes played(x_1) sets per match(x_2) smashes(x_3) successful serve(x_4), back hand shots(x_5), time of play(x_6) Spinning ball shot(x_7).

We can reduce this 7D feature set to 2D feature set. We combine ~~matches~~ x_3, x_2, x_6 to form new feature z_1 game play & x_3, x_4, x_5, x_7 to form over all performance z_2.

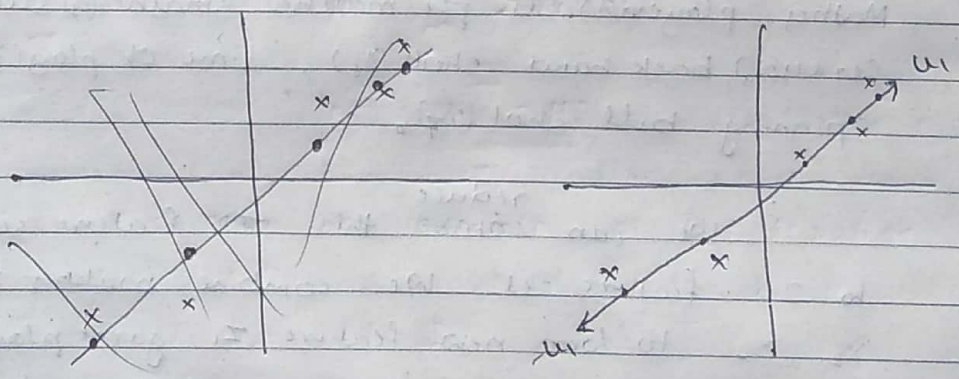


interpret to
we may have kinds of players

a- Experienced & good
b- " but not good
c- Amatear but good
d- " & not good

Principal Component Analysis (PCA)

Let say we have some 2D data & want to reduce it to 1D. hence we need find a best fit line

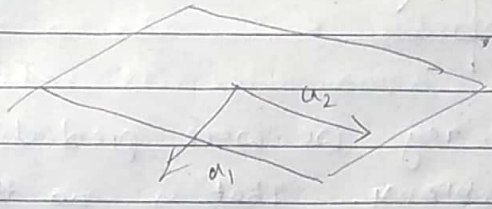


What PCA does is find a lower dimensional line (or vector) which best fits the data i.e has least

projection error

The direction of line u_1 doesn't matter

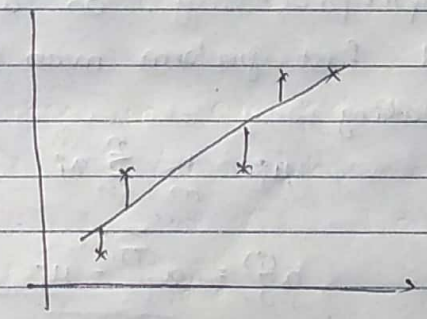
When we are reducing 3D to 2D we should find 2 vectors



Similarly when we reducing to K dimension we need K vectors

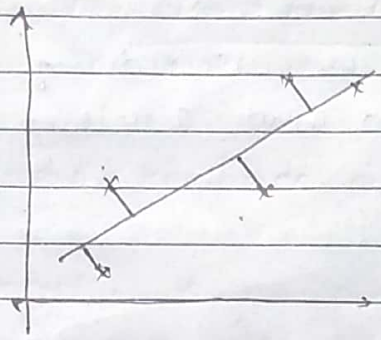
PCA is not linear regression

In linear regression we predict same y for given input feature x_1. So as to minimize the squared distance bet^n predicted & actual values



we do find squared distance that is vertical or $\perp^{lar}$ to x_1 axis

While PCA tries to minimize the squared distance orthogonal to fiting line.



In lin. reg we were predicting some distinct variable y. That is no there in PCA.

IN PCA all variables are treated symmetrically there is no special variable as in the case of lin. reg, the variable y

PCA algorithm.

Step1) → Data preprocessing: on a given training set we should first perform mean normalization or feature scaling.

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{s_j}$$

s_j may be (max-min) or Std. deviation

Step 2) Compute covariance matrix

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} \underbrace{\left(x^{(i)}\right)}_{n \times 1} \underbrace{\left(x^{(i)}\right)^T}_{1 \times n}$$

$n \times n$

Now compute 'eigen vectors' of matrix Σ by U, S, V = Singular Value Decomposition(Σ)

Σ will be $n \times n$ matrix

We need matrix U which is $n \times n$ matrix

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}$$

To reduce n dimension to k dimension we should choose first k columns of U matrix i.e from $u^{(1)}$ to $u^{(k)}$

$\therefore U_{reduced}$ will be $n \times k$ matrix

$$z = (U_{reduced})^T x$$

$$\underbrace{k \times n \quad n \times 1}_{k \times 1}$$

This exclude bias variable x_0.

Choosing the value of k

k is no. of principle components $(z_1, z_2, \cdots z_k)$

Avg. squared proj^n error: $\frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2$

Total variation : $\frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} \|^2$

We typically choose K for which

$$\frac{\text{Avg. squared proj}^{n} \text{ error}}{\text{Total variation}} \leq 0.01 \quad (1\%)$$

i.e 99% variance is retained

Generally K is chosen such that 85-99% of variance is retained

Algorithm

Start with K = 1

for K = 1 to n

Compute $U_{reduce}, z^{(1)}, z^{(2)} \dots z^{(m)}, x_{approx}^{(1)} \dots x_{approx}^{(m)}$

~~Calculate~~

Check if

$$\frac{1/m \sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{1/m \sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \; ?$$

This calculation can be done in a simpler way by using

for given K

$$1 - \frac{\sum_{i=1}^{K} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01 \; ?$$

Where S is matrix S in U, S, V obtained from $U, S, V = svd(\Sigma)$.

S is nxn matrix whose diagonal elements are non-zero rest all will be zero

Now the smallest K that satisfies the

condition is selected

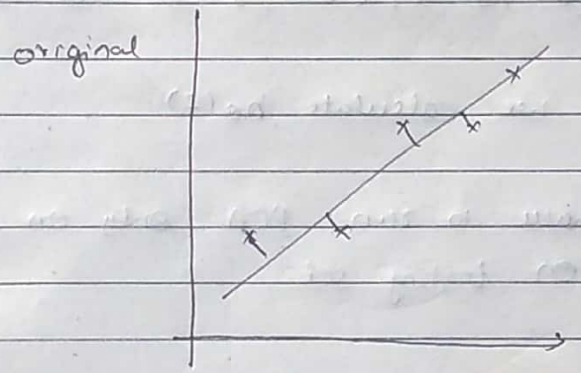
Going back from KD (2D) to nD (3D)

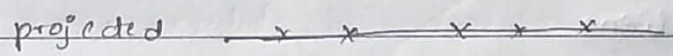
let's say we have

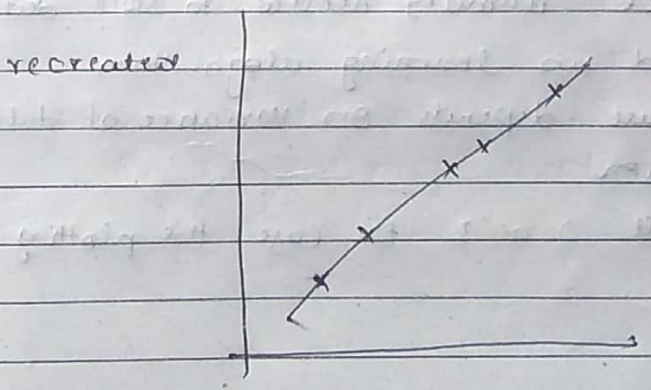
$$z = (U_{reduced})^T x$$

$$x_{approx} = \underbrace{U_{reduced} \times z}_{n \times 1}$$
$$\underbrace{n \times k \times k \times 1}_{n \times 1}$$

As due to PCA projection error is too small x_{approx} will be almost coinciding x







Supervised learning speedup

We have supervised learning problems with dataset $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \ldots (x^{(m)}, y^{(m)})$

Let's say $x^{(i)}$ are very high dimensional

→ We extract the inputs & leave the labels

$x^{(1)} \ldots x^{(m)} \rightarrow$ 10,000 D

→ apply PCA to get

$z^{(1)} \ldots z^{(m)} \rightarrow$ 1000 D

→ new training set will be

$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}) \ldots (z^{(m)}, y^{(m)})$

→ Now we calculate $h_\theta(z)$

→ We have to run PCA only on training & not on testing set

Applications

- Compression
  - reduce memory needed to store data
  - Speed up learning algorithm
    (k value depends on variance of data)
- Visualization
  (k will 2 or 3 to ease the plotting

→ PCA should not be used as solution for overfitting

PCA might work in this regard but it is not a good way to select PCA to address over fitting instead use regularisation

→ PCA might throw out some peice of information

→ Sometimes PCA is used where it shouldn't have been used

hence run learning algorithm without PCA meaning use the original training set if it is not doing what we want then go for PCA