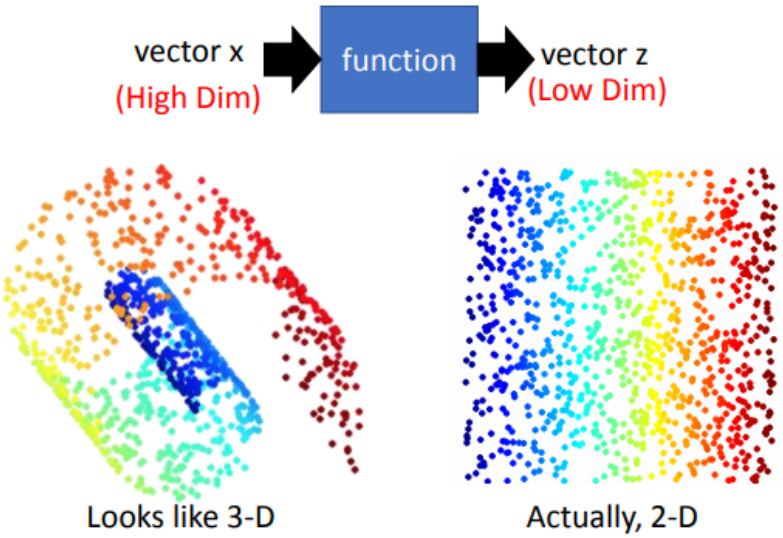
INDEX

Unsupervised Learning

Data 皆不具 label，且訓練時僅有 input 而無法直接獲得 output 的學習模式。主要可分作兩類，Dimension Reduction 與 Generation。

Dimension Reduction

基本精神為「化簡為繁」，意即把本來比較複雜的 input 變成比較簡單的 output。



一、Clustering

假設現在要做 image 的 clustering，那就是把一大堆的 image 分成好幾類。將本來有些不同的 image 都用同一個 class 來表示。

1. K-means

將一大堆的 unlabeled data 把他們分作 K 個 cluster。

首先就是找這些 cluster 的 center，從 training data 裡面隨機找 K 個 object 出來，當成 K 個 cluster 的 center。

接下來決定每一個 object 屬於 1 到 K 的哪一個 cluster。假設現在的 object x^n，跟第 i 個 cluster 的 center 最接近的話，那 x^n 就屬於 c^i。
簡而言之用一個 binary 的 value b （上標 n，下標 i）來代表第 n 個 object 有沒有屬於第 i 個 class，如果第 n 個 object 屬於第 i 個 class 的話，那這一個 binary 的 value 就是 1，反之就是 0。

接下來，就是 update cluster，把所有屬於第 i 個 cluster 的 object 做平均，得到第 i 個 cluster 的 center c^i。

最後就重複上述步驟即可。

- K-means
  - Clustering $X = \{x^1, \cdots, x^n, \cdots, x^N\}$ into K clusters
  - Initialize cluster center c^i, i=1,2, … K (K random x^n from X)
  - Repeat
    - For all x^n in X: $b_i^n \begin{cases} 1 & x^n \text{ is most "close" to } c^i \\ 0 & \text{Otherwise} \end{cases}$
    - Updating all c^i: $c^i = \sum_{x^n} b_i^n x^n \Big/ \sum_{x^n} b_i^n$

2. Hierarchical Agglomerative Clustering (HAC)

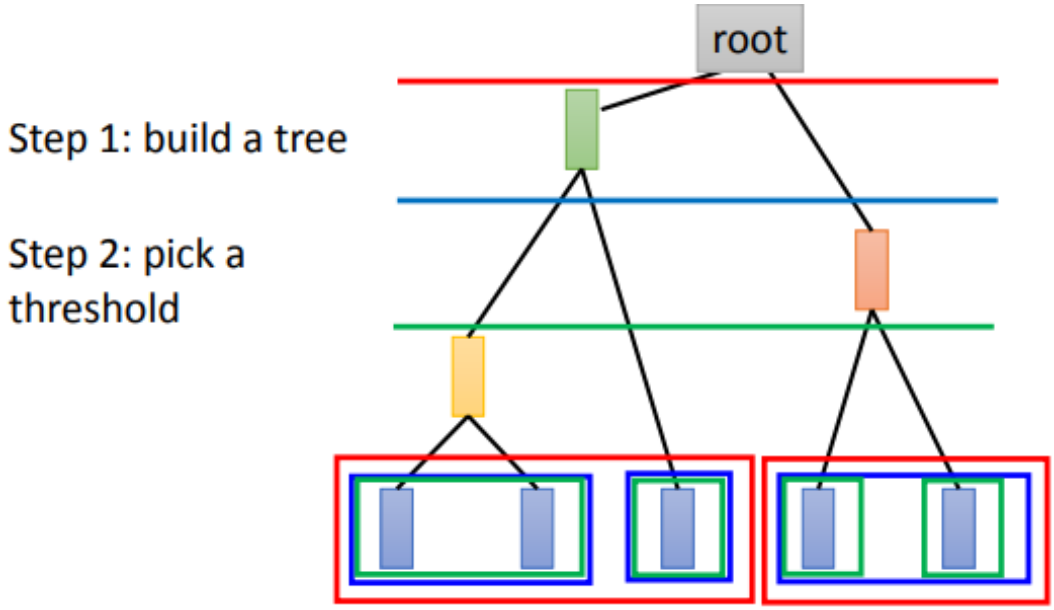
該方法是先建一個 tree，假設現在有 5 個 example，兩兩去算他的相似度，然後挑最相似的那一個 pair 出來。

假設現在最相似的 pair，是第一個和第二個 example，那就把第一個 example 和第二個 example merge 起來，像是對他們取平均得到一個新的 vector （下圖黃色方塊），同時代表第一個和第二個 example。

接下來變成有四個 example，再對這 4 筆 data 兩兩去計算他們的相似度，假設是第三筆和第四筆最像，那就再把他們 merge 起來，得到另外一筆 data（下圖紅色方塊）。

最終得到這個 tree 的 root，建立出一個 tree structure

接下來要決定在這個 tree structure 上面哪地方切一刀，就可將 example 分成好幾個 cluster。



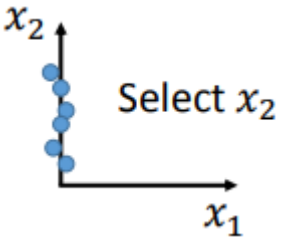
HAC 跟 K-means 最大的差別就是，如何決定 cluster 的數目。在 K-means 裡面要需要決定那個 K 的 value 是多少，而到底有多少個 cluster 是不容易決定的；HAC 的好處就是不直接決定幾個 cluster，而是決定要切在這個樹的 structure 的哪裡。

二、Distributed Representation (Dimension Reduction)

然而在做 cluster 的時候比較以偏概全，因為每一個 object 最後都必須要屬於某一個 cluster。實際上來說應該用一個 vector 來表示各個 object，那這個 vector 裡面的每一個 dimension 就代表了某一種 attribute。
該方式就稱 distributed representation。

1. Feature Selection

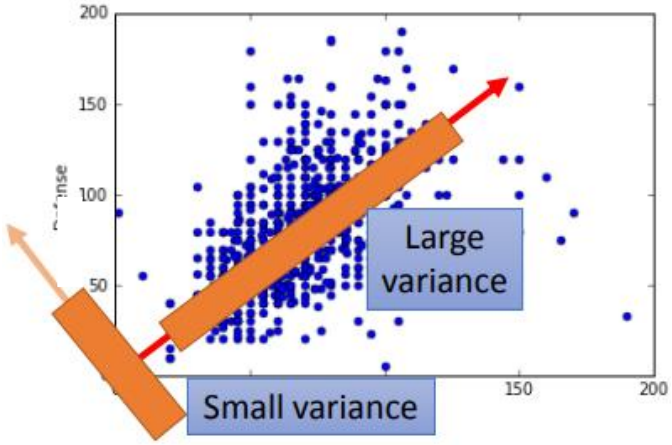
假設 data 的分布本來在二維的平面上，然後發現幾乎都集中在 x_2 的 dimension 而已，如此就可以拿掉 x_1 這個 dimension。
然而這個方法不見得總是有用，因為有很多時候處理的 case 是任何一個 dimension 都不能拿掉的。



2. Principle Component Analysis (PCA)

假設這個 function 是一個很簡單的 linear function，這個 input x 跟這個 output z 之間的關係就是 linear 的 transform，也就是把這個 x 乘上一個 matrix W 可得到 output z。

那現在要做的事情就是根據一大堆的 x 把 W 找出來。可理解成將 x 投影到 W 上，使他們在 W 有較大的 variance，而投影後在 W 上的點就是 z。

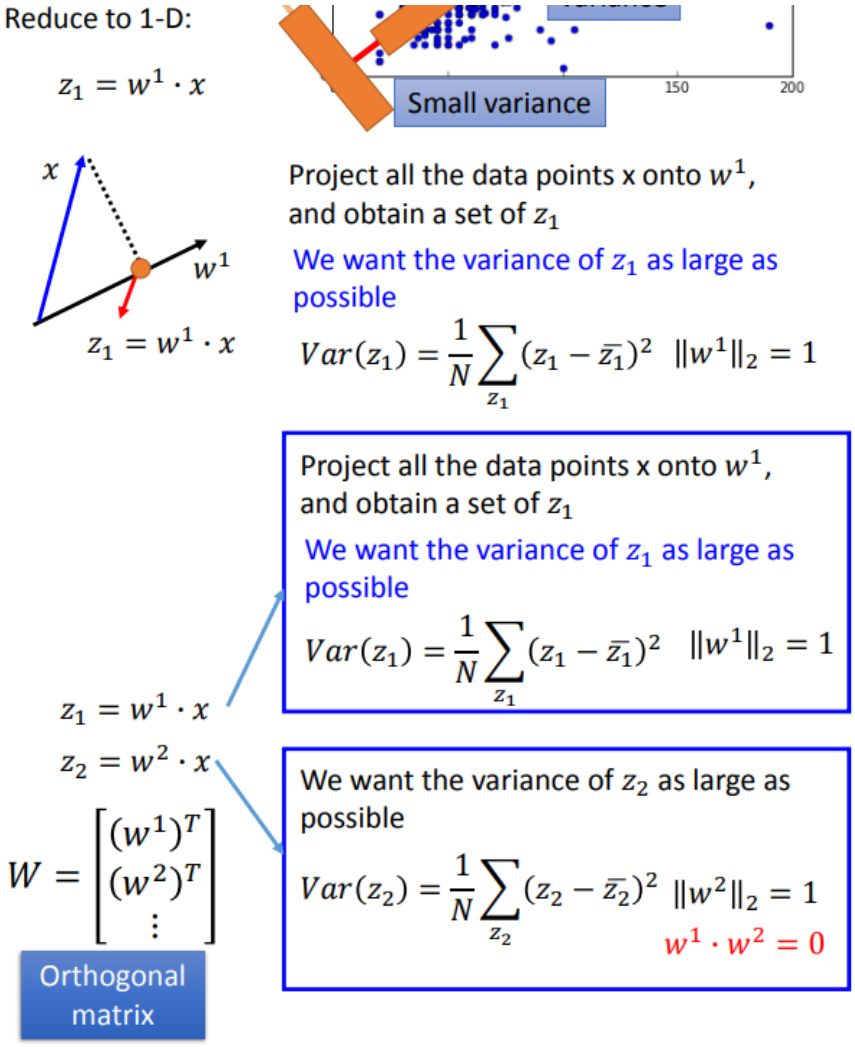


假設把 x 投影到一維，我們希望選一個 w^1，他經過 projection 以後，得到的這些 z1 的分布越大越好。也就是說，我們不希望通過這個 projection 以後，所有的點通通擠在一起（見上圖）。

所以我們希望找一個 projection 的方向，它可以讓 projection 後的 variance 越大越好。因此現在要去 maximize 的對象是 z_1 的 variance，就是 summation over 所有的(z_1 - z_1\bar) 的平方；而 z_1\bar 就是做 z_1 的平均值。因此只要找到一個 w^1 讓 z_1 的 variance 最大就結束了。

再來可能不只要投影到一維，推廣至二維的情況大略相同。同樣是找一個 w^2 讓 z_2 的 variance 最大。

只是在一維時我們必須限制 w^1 的 2-norm，使 w^1 跟 x 做內積能直接得到 z_1；但是為避免 w^2 與 w^1 同值，必須再限制兩者的內積值為 1。



(1) Lagrange Multiplier

前面提到 z_1 等於 w^1 跟 x 的內積值，那 z_1 的平均值就是 summation over 所有 w^1 跟 x 的內積再除以總數。可進一步簡化成 w^1 與 x\bar 的內積，即：

$$\bar{z_1} = w^1 \cdot \frac{1}{N}\sum x = w^1 \cdot \bar{x}$$

而 z_1 的 variance 可整理為 w^1 的 transpose 乘上 x 的 covariance 再乘上 w^1。
而此處用 S 來描述 x 的 covariance matrix。

所以現在要解的問題是找出一個 w^1 可以 maximize 該式。但這個 optimization 的對象是有 constraint 的，如果沒有 constraint 的話，這裡的 w^1 每一個值都變無窮大就結束了。所以這裡的 constraint 是說 w^1 的 2-norm 要等於 1。

$$Var(z_1) = \frac{1}{N}\sum_{z_1}(z_1 - \bar{z_1})^2$$

$$= \frac{1}{N}\sum_{x}(w^1 \cdot x - w^1 \cdot \bar{x})^2$$

$$= \frac{1}{N}\sum\left(w^1 \cdot (x - \bar{x})\right)^2$$

$$= \frac{1}{N}\sum (w^1)^T (x - \bar{x})(x - \bar{x})^T w^1$$

$$= (w^1)^T \boxed{\frac{1}{N}\sum (x - \bar{x})(x - \bar{x})^T} w^1$$

$$= (w^1)^T Cov(x) w^1 \qquad S = Cov(x)$$

$$(a \cdot b)^2 = (a^T b)^2 = a^T b a^T b$$

$$= a^T b (a^T b)^T = a^T b b^T a$$

Find w^1 maximizing

$$(w^1)^T S w^1$$

$$\|w^1\|_2 = (w^1)^T w^1 = 1$$

那有了這些以後，我們就要解這一個 optimization 的 problem。
由於 S 是 symmetric 又是 positive-semidefinite 的關係，他所有的 eigenvalue 都是 non-negative 的。

接著用 Lagrange multiplier（開頭如下式假設），

$$g(w^1) = (w^1)^T S w^1 - \alpha((w^1)^T w^1 - 1)$$

把這個 function 對 w 的第一個 element 做偏微分，再對第二個 element 做偏微分，依此類推。然後令這些式子通通等於 0，整理完後得到，會得到一個式子帶入 w^1 後使其為 0。

$$S(w^1) - \alpha(w^1) = 0$$

而 w^1 就是 S 的 eigenvector，接下來看哪一個 eigenvector 代到下式，可以 maximize 該式。

$$(w^1)^T S(w^1) = \alpha(w^1)^T(w^1) = \alpha$$

所以問題變成找一個 w^1 使 α 最大。而當 α 最大時，這個 α 就是最大的 eigenvalues λ_1；w^1 是對應到最大的 eigenvalue 的 eigenvector。

Find w^1 maximizing $(w^1)^T S w^1$ $(w^1)^T w^1 = 1$

$S = Cov(x)$ Symmetric Positive-semidefinite (non-negative eigenvalues)

Using Lagrange multiplier [Bishop, Appendix E]

$$g(w^1) = (w^1)^T S w^1 - \alpha\left((w^1)^T w^1 - 1\right)$$

$\partial g(w^1)/\partial w_1^1 = 0$
$\partial g(w^1)/\partial w_2^1 = 0$
$\vdots$

$Sw^1 - \alpha w^1 = 0$
$Sw^1 = \alpha w^1$ w^1 : eigenvector

$(w^1)^T S w^1 = \alpha(w^1)^T w^1$
$= \alpha$ Choose the maximum one

w^1 is the eigenvector of the covariance matrix S
Corresponding to the largest eigenvalue λ_1

同理，如果要找 w^2 的話，就要 maximize 根據 w^2 投影以後的 variance：

$$(\mathrm{w}^2)^{\mathrm{T}}\mathrm{S}(\mathrm{w}^2)$$

同樣假設 function g 裡面包含了你要 maximize 的對象，還有兩個 constraint（w^1 跟 w^2 他們是 orthogonal 的），然後分別乘上 α 跟 β。

$$g(w^2) = (\mathrm{w}^2)^{\mathrm{T}}\mathrm{S}(\mathrm{w}^2) - \alpha((w^2)^T w^2 - 1) - \beta((w^2)^T w^1 - 0)$$

接下來對所有的參數做偏微分得到這個值：

$$S(w^2) - \alpha(w^2) - \beta(w^1) = 0$$

接著式子左邊同乘 w^1 的 transpose 變為：

$$(w^1)^T S(w^2) - \alpha(w^1)^T(w^2) - \beta(w^1)^T(w^1) = 0$$

紅字部分為一個 scalar (vector* matrix* vector)，而 scalar 在做 transpose 以後還是他自己，所以 transpose 結果是一樣的，得到：

$$(w^1)^T S(w^2) = (w^2)^T(S^T)(w^1) = (w^2)^T S(w^1)$$

（因為 S 是 symmetric 的，所以 transpose 以後還是他自己）

接下來我們已經知道 w^1 是 S 的 eigenvector，而且它對應到最大的 eigenvalue λ^1，所以寫為下式：

$$\because S(w^1) = (\lambda^1)(w^1)$$

$$\therefore (w^2)^T S(w^1) = (w^2)^T(\lambda^1)(w^1) = (\lambda^1)(w^1)(w^2)^T$$

因為$(w^1)*(w^2)^T$ 又等於 0 (orthogonal)，所以得到的結論是如果 β 等於 0 的話，剩下的 $S*(w^2)$會等於 $\alpha*(w^2)$。

所以 w^2 也是一個 eigenvector 且必須跟 w^1 orthogonal，故選第二大的 w^2，然後他對應到第二大的 eigenvalue λ^2。
其餘維度依此類推。

Find w^2 maximizing $(w^2)^T S w^2$ $(w^2)^T w^2 = 1$ $(w^2)^T w^1 = 0$

$$g(w^2) = (w^2)^T S w^2 - \alpha\left((w^2)^T w^2 - 1\right) - \beta\left((w^2)^T w^1 - 0\right)$$

$\partial g(w^2)/\partial w_1^2 = 0$

$\partial g(w^2)/\partial w_2^2 = 0$

$\vdots$

$$Sw^2 - \alpha w^2 - \beta w^1 = 0$$

$$0 - \alpha\, 0 - \beta\, 1 = 0$$

$$= \left((w^1)^T S w^2\right)^T = (w^2)^T S^T w^1$$

$$= (w^2)^T S w^1 = \lambda_1 (w^2)^T w^1 = 0$$

$$Sw^1 = \lambda_1 w^1$$

$\beta = 0$: $Sw^2 - \alpha w^2 = 0$ $Sw^2 = \alpha w^2$

w^2 is the eigenvector of the covariance matrix S
Corresponding to the 2nd largest eigenvalue λ_2

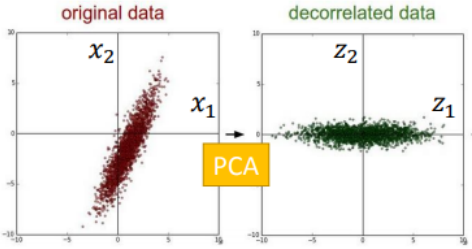
另外 PCA 中 z 的 covariance 會是一個 diagonal matrix。
也就是說，假設 PCA 所得到的新的 feature z 給其他的 model 描述某一個 class 的 distribution （假設為 generative model）。
那在做這個 Gaussian 的假設的時候，假設說 input data 的 covariance 就是 diagonal，且不同的 dimension 之間沒有 correlation，這樣一來減少參數量。
所以他就可以用比較簡單的 model 來處理 input data，避免 overfitting 的情形

PCA - decorrelation

$z = Wx$

$Cov(z) = D$

Diagonal matrix



$$Cov(z) = \frac{1}{N}\sum (z - \bar{z})(z - \bar{z})^T = WSW^T \quad S = Cov(x)$$

$$= WS[w^1 \quad \cdots \quad w^K] = W[Sw^1 \quad \cdots \quad Sw^K]$$

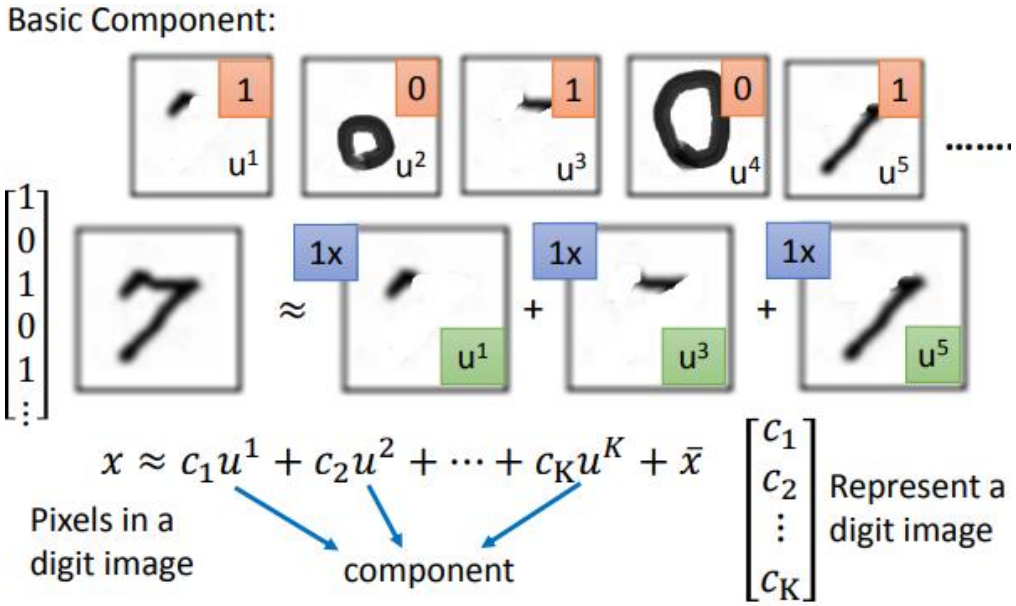
$$= W[\lambda_1 w^1 \quad \cdots \quad \lambda_K w^K] = [\lambda_1 W w^1 \quad \cdots \quad \lambda_K W w^K]$$

$$= [\lambda_1 e_1 \quad \cdots \quad \lambda_K e_K] = D$$ Diagonal matrix

(2) SVD

假設在考慮的是 MNIST，這些數字其實是由一些 basic 的 component（筆畫）所組成的。那這些 component 寫作 u^1, u^2, u^3 等等。則 input x 會等於 u^1 這個 component 乘上 c^1 加上 u^2 這個 component 乘上 c^2，以此類推，然後再加上 x\bar 代表所有的 image 的平均。

所以每一張 image 就是有一堆 component 的 linear combination，然後再加上它的平均所組成的。



接著這一些 linear combination 的結果減去 x\bar，該值必須與目標值 x\head 越近越好，即：

$$x - \bar{x} \approx c_1u^1 + c_2u^2 + \cdots + c_Ku^K = \hat{x}$$

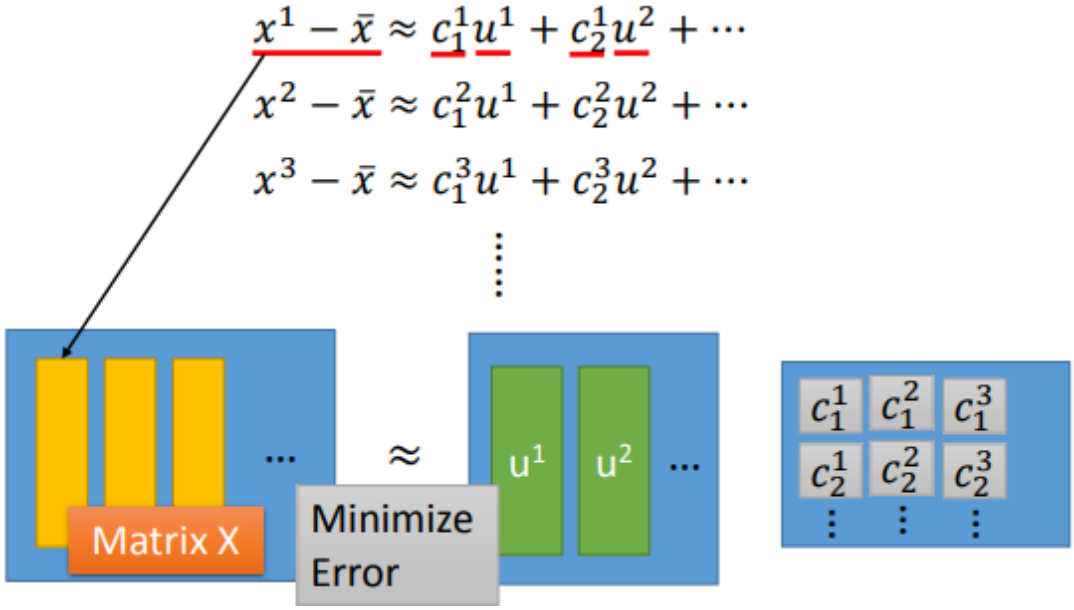
因此就必須找 K 個 vector 去 minimize 他們的距離 (reconstruction error)

Reconstruction error：$\|(x - \bar{x}) - \hat{x}\|_2$

$$L = \min_{\{u^1,\ldots,u^K\}} \sum \|(x - \bar{x}) - \hat{x}\|_2$$

$$\hat{x} = \sum_{k=1}^{K} c_k u^k$$

接下來，可進一步將 reconstruction error 表示為 matrix 的乘積。



接著可以用 SVD 把 matrix X 拆成 U，Σ 與 V 三個 matrix 的乘積，U 就是代表 matrix u^K；Σ*V 就是代表 matrix c_K。

然後 U 這個 matrix，他的 k 個 column，其實就是一組 orthonormal vector，對應到的就是 $X*X^T$ 最大的 k 個的 eigenvector。

而這個 $X*(X^T)$就是 covariance matrix，也就是 PCA 找出來的那一些 w（covariance matrix 的 eigenvector）等同於解出來的 U 的每一個 column 的 vector。

換句話說，根據 PCA 找出來的那些 w 其實就是在 minimize 這個 reconstruction error；那 Dimension Reduction 的結果就是這些 vector。

(3) Neural Network
已知從用 PCA 找出來的 w^1 到 w^K 就是 K 個 component，u^1, u^2 到 u^K，再根據 component linear combination 得到的結果叫做 x\head，也就是$(w^K)*c_k$ 做 linear combination 的結果。

那我們會希望說，這個 x\head

跟 (x - x\bar)

(x - x\bar)，它的平均的是越小越好

你要 minimize 這個 Reconstruction error

那我們現在已經根據 SVD

找出來的 W，W 已經找出來了

W 已經找出來了，那 ck 的值到底應該是多少呢？

這個 ck 是每一個 example

如果是 image recognition 的話，就每一個 image

都有一組自己的 ck

所以，你要找這個 ck 就每一個 image 各自找就好

每一個 image 各自找就好了

那這個問題，其實就是問說

我現在有 K 維的 vector

它們做 span 以後，得到一個 space

如果我現在

要用 c1 到 cK 對它做 linear combination

怎麼樣才能夠最接 (x-x\bar)

怎麼樣才能夠最接 (x-x\bar) 呢

因為現在這 K 個 vector 它們是 orthonormal 的

所以你要得到這個 ck，其實是很簡單的，你只要把

(x-x\bar) 跟 w^k 做 inner product

你要找一組 ck 可以

那這個性質是來自於說

minimize 左邊這個跟右邊這個的 error

你只需要把

這個 (x-x\bar) 跟 w^k 做 inner product 就好了

那這個性質是來自於說

這 K 個 vector 是 orthonormal 的

這個如果你有困惑的話

就回去 check 一下線性代數的課本

那我們現在已經知道了這些事情

我們已經知道說

c^k 就是長成這個樣子

那這件事情呢

這個做 linear combination 的事情

其實你可以想成用 neural network 來表示它

什麼意思呢？

假設我們的 (x-x\bar) 就是一個 vector

這邊寫作一個三維的 vector

那我們假設，現在 K 只有兩個 component

K 等於 2

那我們先算出 c^1 跟 c^2

怎麼算 c^1 呢？

c^1 就是 (x-x\bar) 跟 w^1 的 inner product

所謂的 inner product 就是 element-wise 的相乘

也就是把 (x-x\bar) 的每一個 component

乘上 w^1 的每一個 component

接下來，你就得到 c^1

這件事情就好像是說

這個是 neural network 的 input

這是一個 neuron，這是 neuron 的 weight

這個 neuron 它是 linear 的 neuron

它沒有 activation function，它是 linear 的

那這個 neuron，你把這個東西 input

乘上這個 weight，你就得到 c^1

那 c^2 也是一樣

c^2，我們這邊

這邊是這樣子，那我們接下來要把 c^1

我這邊犯了一個錯

這個應該是下標

這個應該是下標

如果統一起來的話，這個 K 應該是下標

那我們把這個 c1 乘上 w^1

所謂的 c1*(w^1)是什麼意思呢？

你就把 c1*(w^1)

把 c1 乘上 w^1 的第一維，得到一個 value

乘上 w^1 的第二維，得到一個 value；
乘上 w^1 的第三維，得到一個 value

這一項，就是 c1*(w^1)

接下來，我們再算一下 c^2

c^2 一樣就是這個 input 一樣乘上

跟這個 w^2 做 inner product

得到 c^2，然後再把這個 c^2

w^2 的三個 element

再跟原來 w^1 的三個 element 加起來

得到最後的 output，這一項就是 x\head

這一項就是 x\head

接下來，我們 training 的 criteria

就是 minimize

我們要讓這個 x\head 跟 (x-x\bar) 越接近越好

你所以，我們就是希望這個 neural network 的 output

跟 (x-x\bar) 越接近越好

這是我們的 input, (x-x\bar)

它乘上一組 weight

再 hidden layer 的 output 是 c^1, c^2

再乘上另外一組 weight，得到 x\head

那我們希望 (x-x\bar) 越接近越好

那你就會發現說

其實 PCA 可以表示成一個 neural network

它可以表示成一個 neural network，然後

這個 neural network 它只有一個 hidden layer

然後，這個 hidden layer 是 linear 的 activation function

然後，我們現在 train 這個 neural network 的 criterion

是要讓 input 一個東西，得到 output

結果這個 output 要跟 input 越接近越好

這個 output 要跟 input 越接近越好

這個 output 要跟 input 越接近越好

這個東西就叫做 Autoencoder