**Slide-1**

**Dimension Reduction**

****

**Slide-2**

* **PCA – Principal Component analysis**
* **SVD – Single value Decomposition**

**Slide-3**

**Dimension Reduction – Applications**

* Computational performance enhanced
* Face recognition
* Image compression

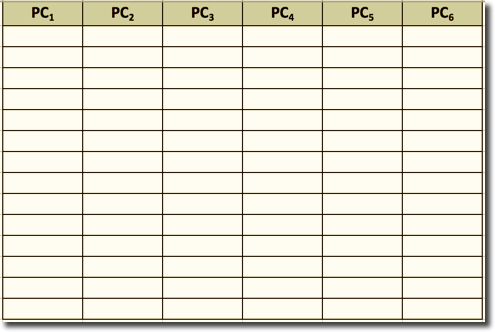
**Slide-4**

**PCA – Benefits**

* Reduce number of columns
* Identify relation between columns
* Visualize multidimensional data in 2D

**Slide-5**

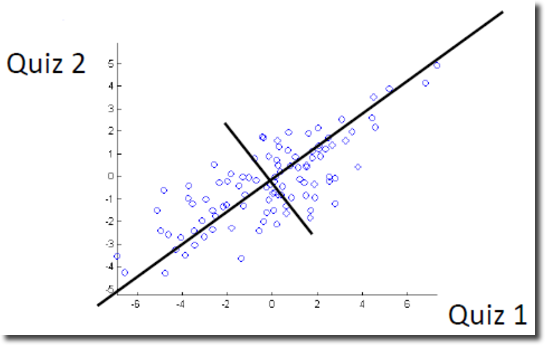
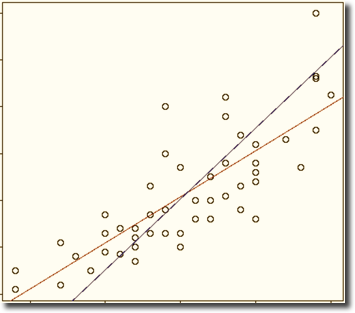
**PCA key Benefits**

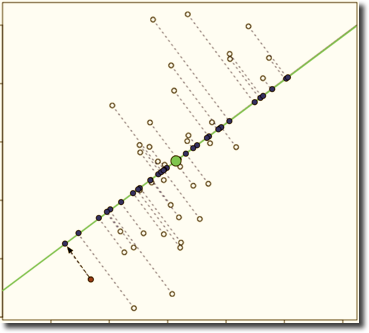
** **

Less number of columns will capture maximum information

**Slide-6**

**PCA Intuition**

** **

****

Reduces the number of columns and captures the maximum information of the entire data set.

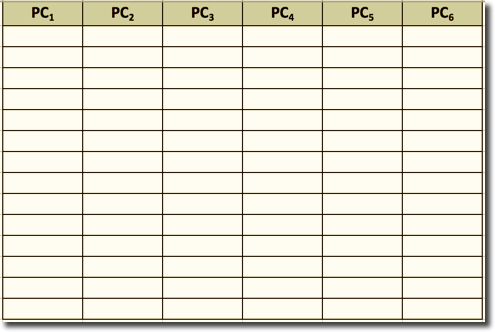
**SLIDE-7**

**PCA Preliminaries**

|  |  |
| --- | --- |
| **Input (Before PCA)**   * ***n’ original columns*** | **Output (After PCA)**   * ***n’ principal components (‘n’ weighted averages of original measurements)*** |
| **Input (Before PCA)**   * ***Correlated*** | **Output (After PCA)**   * + ***Uncorrelated***   + ***Ordered by variance***   + ***Keep top principal components; drop rest*** |

**SLIDE-8**

PCA WEIGHTS:

**** ****

The ith principal component is a weighted average of original measurements / columns:

PCi = ai1X1+ ai2 X2+…..+ aipXp

Weights (aij) are chosen such that:

* 1. PCs are ordered by their variance (PC1 has largest variance, followed by PC2, PC3, and so on)
  2. Pairs of PCs have correlation = 0
  3. For each PC, sum of squared weights = 1 (Unit Vector)

**SLIDE-9**

**PCA Weights**

**PCi = ai1X1 + ai2X2 + ai3X3…. + ainXn**

Main idea: High variance = lots of information

*Var (PCi) = a2i1 Var(X1) + a2i2 Var(X2) + a2i3 Var(X3) + … + a2in Var(Xn) + 2ai1ai2 Cov(X1, X2) + … + 2ain-1ain Cov(Xn-1, Xn)*

*Also, Covar(PCi , PCj) = 0, when i* ≠ j

* Goal 1: Find weights aij that maximize variance of PCi, while keeping PCi uncorrelated to other PCs
* The covariance matrix of the Xs is needed
* PCs are uncorrelated
* Var(PC1) > Var(PC2) > Var(PC3) > Var(PC4) >…..........

**SLIDE-10**

Standardize the variables:

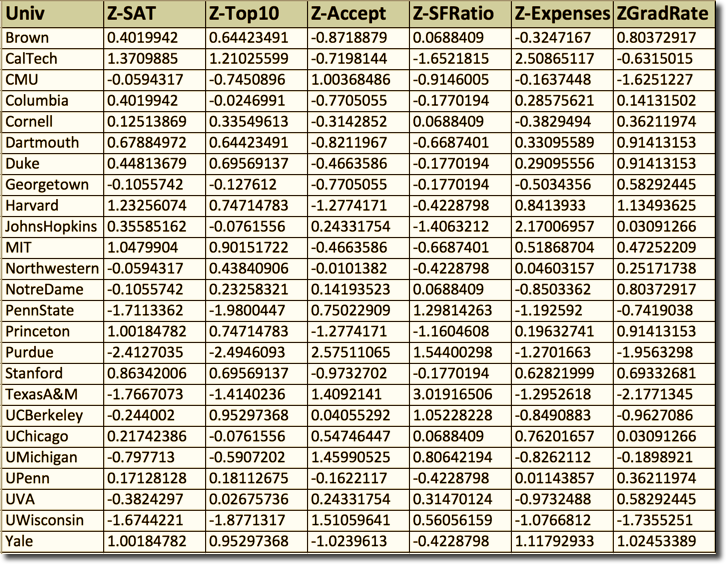
Why?

We standardize because variables with large variances will have bigger influence on result

We standardize before applying PCA

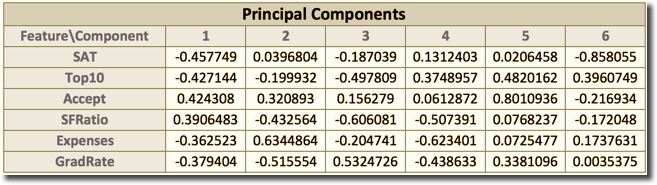
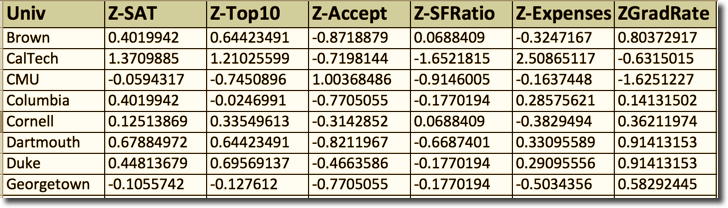
Correlation matrix is the matrix which standardizes the data before finding the correlation values

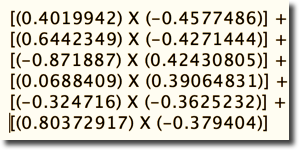
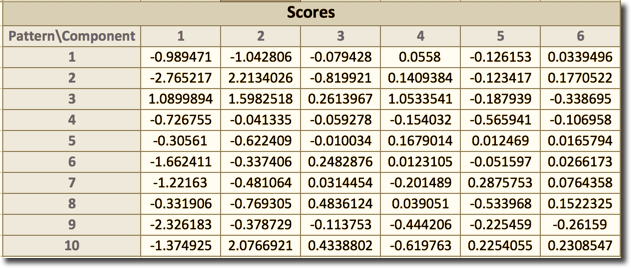
Covariance matrix is the matrix, which does not standardize the data before finding the correlation values



SLIDE-11

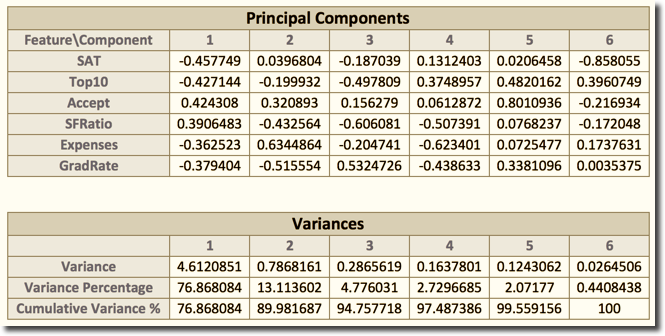
PCA CALCULATION

**SLIDE-12**

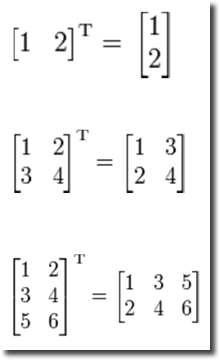
**PCA Goal 1 – Data Reduction**



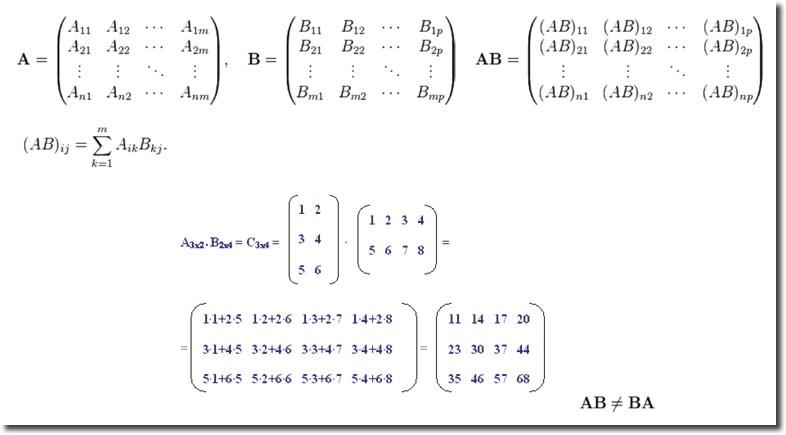
* PCs are ordered by variance
* PC1 captures 76.86% of the information
* First 2 PCs capture 89.98% of the information
* Choose first few PCs & drop the rest

SLIDE-13

**PCA Goal 1 – Data Compression – Matrix Basics**

****

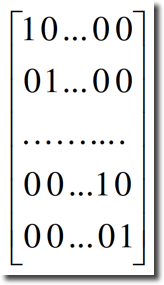
**SLIDE-14**

****

**SLIDE-15**

If A x B = I, identity matrix then B = A-1

Identity Matrix:



**SLIDE-16**

**PCA Goal 1 – Data Compression**

[PCScores]Nxp  = [ScaledData]Nxp x [PrincipalComponents]pxp

[ScaledData]Nxp = [PCScores]Nxp  x [PrincipalComponents]-1pxp

= [PCScores]Nxp  x [PrincipalComponents]Tpxp

Approximation:

[ApproximatedScaledData]Nxp = [PCScore]Nxc X [PrincipalComponent]Tcxp

Note: c = Number of components kept; c <= p

SLIDE-17

**PCA Goal 2 – Labeling PCs**

Learn relationships with PCA by interpreting the weights

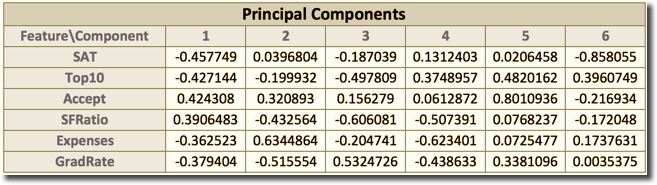
• ai1,…, aip are the coefficients for PCi.

• They describe the role of original X variables in computing PCi.

• Useful in providing context-specific interpretation of each PC.

PC1 weights measure the degree of:

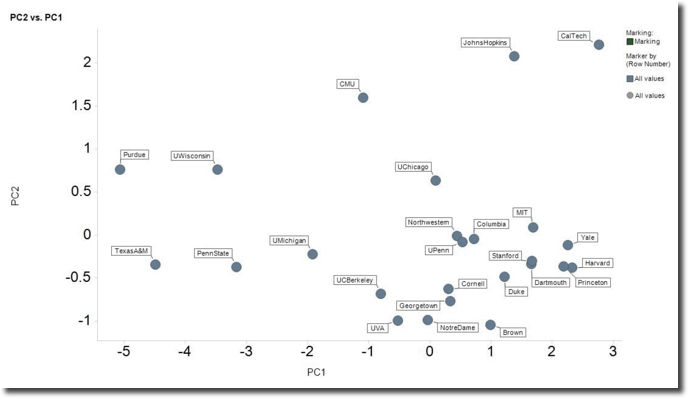
* + High Accept, SFRatio
  + Low Expenses, GradRate, SAT, Top10



SLIDE-18

**PCA Goal 3 – PCA for visualization**

* The first 2 or 3 PCs can be used to project data from an n-dimensional space onto a 2D or 3D space



**SLIDE-19**

**PCA – Additional Benefits**

Monitoring batch processes

using PCA

• Multivariate data at different time points

• Historical database of successful batches are used

• Multivariate trajectory data is projected to low-dimensional space

>>> Simple monitoring charts to spot outlier

**SLIDE-20**

**PCA – Give a thought!**

* If we use a subset of the principle components, is this useful for prediction? Is it useful for explanation?
* What are the pros & cons of PCA compared to choosing a subset of the variables?
* Differences between PCA & Clustering?

**SLIDE -21**

**SVD – Singular Value Decomposition**

**Original size = Compressed size = 15,380 bytes 147,456 bytes**

** **

Source: http://journal.batard.info/post/2009/04/08/svd-fun-profit

SLIDE-22

**SVD – Preliminaries**

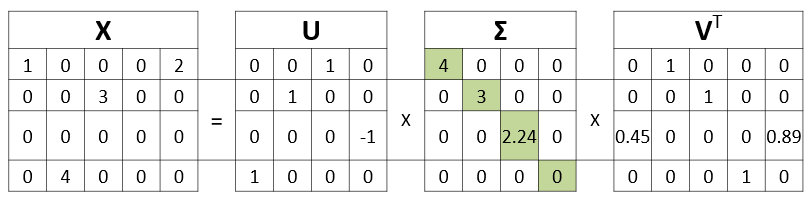
X = UΣVT

* U & V are Orthonormal matrices
* Orthonormal = Orthogonal + Unit Length (Normalize + Sum)

**N x n N x r r x r r x n**

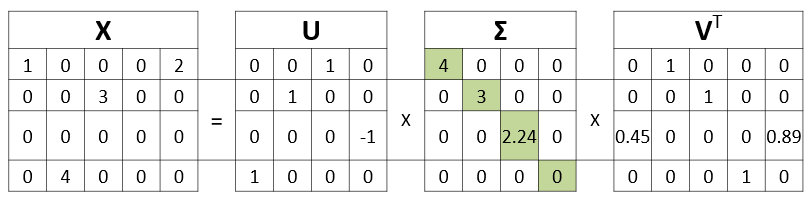
= X X

SLIDE-23



* Xk = i iT
* Each term in the summation expression provided above is called ***Principle Image***

SLIDE-24



Original Size = 4\*5 = 20 bytes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **k = 1** | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | x | 4 | x | 0 | 1 | 0 | 0 | 0 | = | 0 | 0 | 0 | 0 | 0 |
| 0 |  |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 | 0 |
| 0 |  |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 | 0 |
| 1 |  |  |  |  |  |  |  |  |  | 0 | 4 | 0 | 0 | 0 |

***Compressed Size = 4\*1 + 1 + 1\*5 = 10 bytes***

**SLIDE-25**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **k = 2** | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | x | 4 | 0 | x | 0 | 1 | 0 | 0 | 0 | = | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 |  | 0 | 3 |  | 0 | 0 | 1 | 0 | 0 |  | 0 | 0 | 3 | 0 | 0 |
| 0 | 0 |  |  |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 |  |  |  |  |  |  |  |  |  |  | 0 | 4 | 0 | 0 | 0 |

***Compressed Size = 4\*2 + 2 + 2\*5 = 20 bytes***

• Original size = 384\*384 bytes = 147,456

bytes

• k=1: 384\*1+1+1\*384=769 bytes

• k=10: 384\*10+10+10\*384=7,690 bytes

• k=20: 384\*20+20+20\*384=15,380 bytes

• k=50: 384\*50+50+50\*384=38,450 bytes

• k=100: 384\*100+100+100\*384=76,900 bytes

• k=200: 384\*200+200+200\*384=153,800 bytes