DENSE VECTORS

MATRIX DECOMPOSITION

$\overrightarrow{\text{apple}} \sim \{\overrightarrow{\text{iphone}}, \overrightarrow{\text{ipad}}, \overrightarrow{\text{imac}}, \ldots\}$

$\overrightarrow{\text{apple}} - \overrightarrow{\text{iphone}} \sim \{\overrightarrow{\text{raisin}}, \overrightarrow{\text{pecan}}, \overrightarrow{\text{cranberry}} \ldots\}$

$\overrightarrow{\text{virus}} \sim \{\overrightarrow{\text{malware}}, \overrightarrow{\text{infection}}, \overrightarrow{\text{infected}} \ldots\}$

$\overrightarrow{\text{virus}} - \overrightarrow{\text{malware}} \sim \{\overrightarrow{\text{hepatitis}}, \overrightarrow{\text{herpes}} \ldots\}$

$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} \sim \overrightarrow{\text{queen}} - \overrightarrow{\text{woman}}$

$\overrightarrow{\text{animal}} - \overrightarrow{\text{cat}} \sim \overrightarrow{\text{animal}} - \overrightarrow{\text{dog}}$

Ramaseshan

PATTERNS IN DATA

- Given a large sets of data, we wish to observe trends and patterns in larger data
- Find out if there are any relationships among variables
- To observe the pattern visually for a vocabulary of size $n \times n$, imagine plotting the m coordinates representing the n vocabulary in n dimensional space.
- The similar correlated words are clustered in this space – hard to imagine
- We need to recognize n orthogonal axes fro the data set – identifying the
- Manual identification of relationship is not feasible, as the task is $O(n^2)$, if n is the size of the data

GOAL

Map the high dimensional samples in R^m to a lower dimensional space R^k, where $k \ll m$, ideally preserving the the local and global structure of the original data

Is high dimensional data truly lower dimensional?

WHY DENSE VECTOR?

- Assumption: Data lies in the lower dimensional space
- Input data may have 100K+ dimensions
  - COVID19 data set has 227 Billion tokens no preprocessing
  - Word vector size - 3.5 Million
  - Dimensionality reduction: can we compress Word vector from $R^{3.5M}$ into R^{100}?
- Fewer dimensions
  - fewer parameter to fine-tune and fast training
  - Yet discover latent relationship of data

SYMMETRIC WORD-WORD CO-OCCURRENCE MATRIX

If the matrix is real symmetric, then it can be diagonalized

$$S = Q\Lambda Q^T$$

Orthogonal eigen vectors

Real eigen values

Are symmetric word-word co-occurrence matrices positive semi-definite?

SINGULAR VALUE DECOMPOSITION

SINGULAR VALUE DECOMPOSITION

Co-occurrent matrix A is not quite square matrix but symmetrical

AA^T and A^TA are symmetrical and real, but may not be equal

$$AA^Tu_i = \sigma_i^2 u_i$$
$$A^TAv_i = \sigma_i^2 v_i$$
$$Av_i = \sigma_i u_i$$

$$X = U\Sigma V^T = u_1\sigma_1 v_1 + u_1\sigma_1 v_1 + \cdots + u_k\sigma_k v_k$$
$$\sigma_1 > \sigma_2 > \cdots > \sigma_k$$

$u_i u_j^T = \delta_{ij}$ and $v_i v_j^T = \delta_{ij}$

$$\delta_{ij} = \begin{cases} = 1, \text{ if } i = j \\ = 0, \text{ if } i \neq j \end{cases}$$

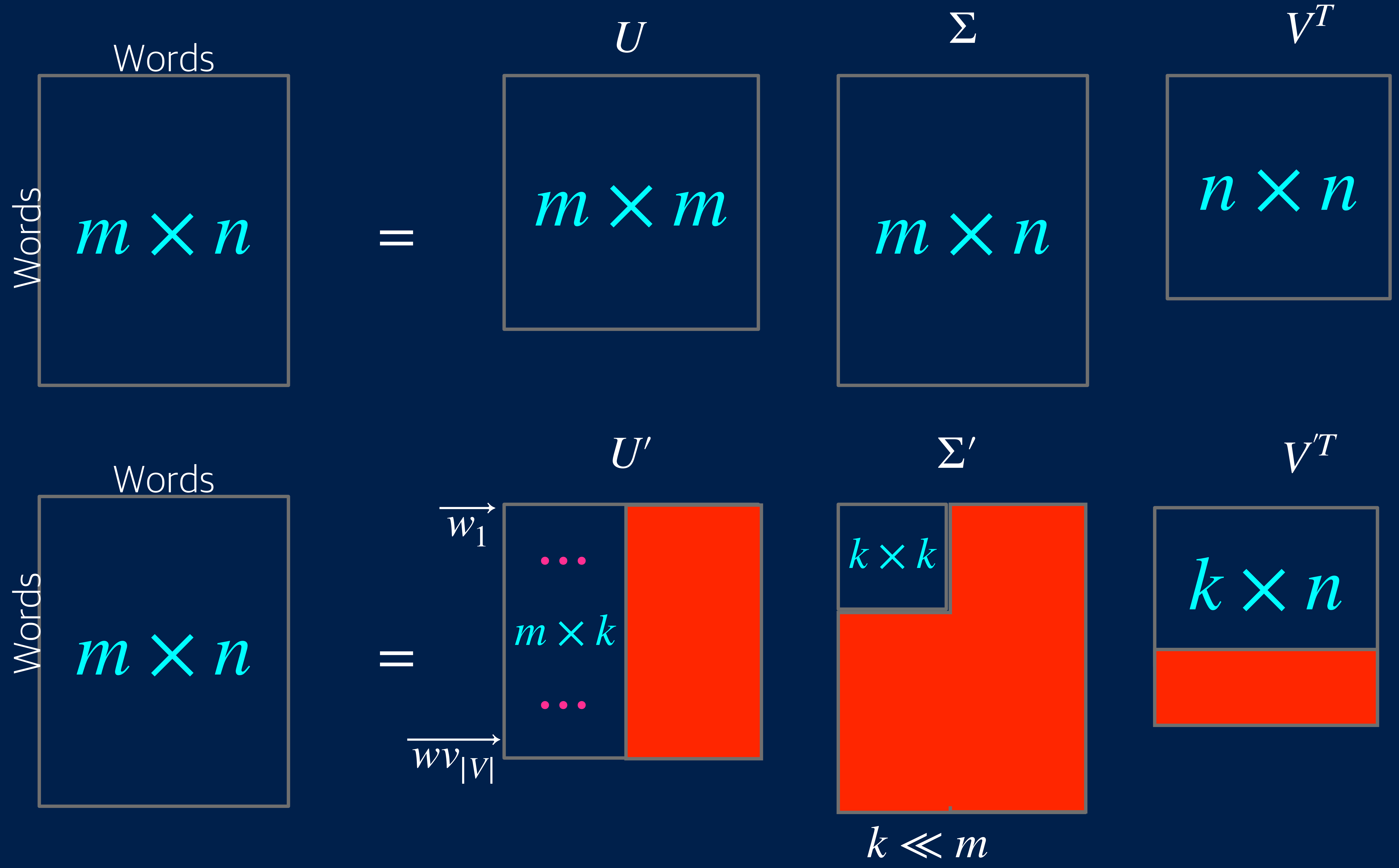
SVD breaks down the original relationships into linearly independent components

LATENT SEMANTIC ANALYSIS [DEERWESTER ET AL]

SVD

- In any natural language, it is reasonable to assume that the words can be clustered or grouped into latent factors (L) that
  - Share similar connectivity or correlation patterns
  - Transitive relationship through n-order association
- Many singular values are very small to have any substantial impact
  - Result in $|L| \ll |V|$
- Lead to an approximate model with fewer dimensions
- w_i and wj similarity is now approximated using smaller number of dimensions
- Dot product or cosine between vectors represent their estimated similarity
- $\| X - \hat{X} \| \neq 0$ where $X = AA^T$

CHOICE OF k



k –large enough to fit all the latent (hard to measure directly)

Patterns in the data and small enough to omit unimportant details

PRINCIPAL COMPONENT ANALYSIS

The goal is to express the data set into another basis - a linear combination of the original basis to re-expresses the original data set

SEMANTIC FACTOR ANALYSIS

Let $x_i \in \mathbb{R}^M$ be the set of observations (cooccurrence values) from

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{M \times N}$$

- Let X and Y be $m \times n$ matrices related by a linear transformation P.
- X is the original co-occurrent data set and Y is a re-representation of that data set.
- $Y = PX$
- Geometrically, P rotates, stretches and transforms X into Y
- Express the word vectors as a linear combination of latent factors

PCA

- Choose an orthonormal matrix P where $Y = PX$ such that $S_Y \equiv \frac{1}{N-1} YY^T$ is diagonalized.
- The rows of P are the principal components of X
- The i^{th} diagonal value of S_Y is the variance of X along p_i
- Note: X is a zero-mean data $X = X - \bar{X}$

A tutorial on PCA

$$S_y = \frac{1}{N-1} PX(PX)^T$$
$$= \frac{1}{N-1} PXX^T P^T$$
$$= \frac{1}{N-1} PAP^T$$
$$A = Q\Lambda Q^T$$

A is symmetric and is diagonalizable.

Now,

$$S_Y = \frac{1}{n-1} P(Q\Lambda Q^T)P^T -$$ here

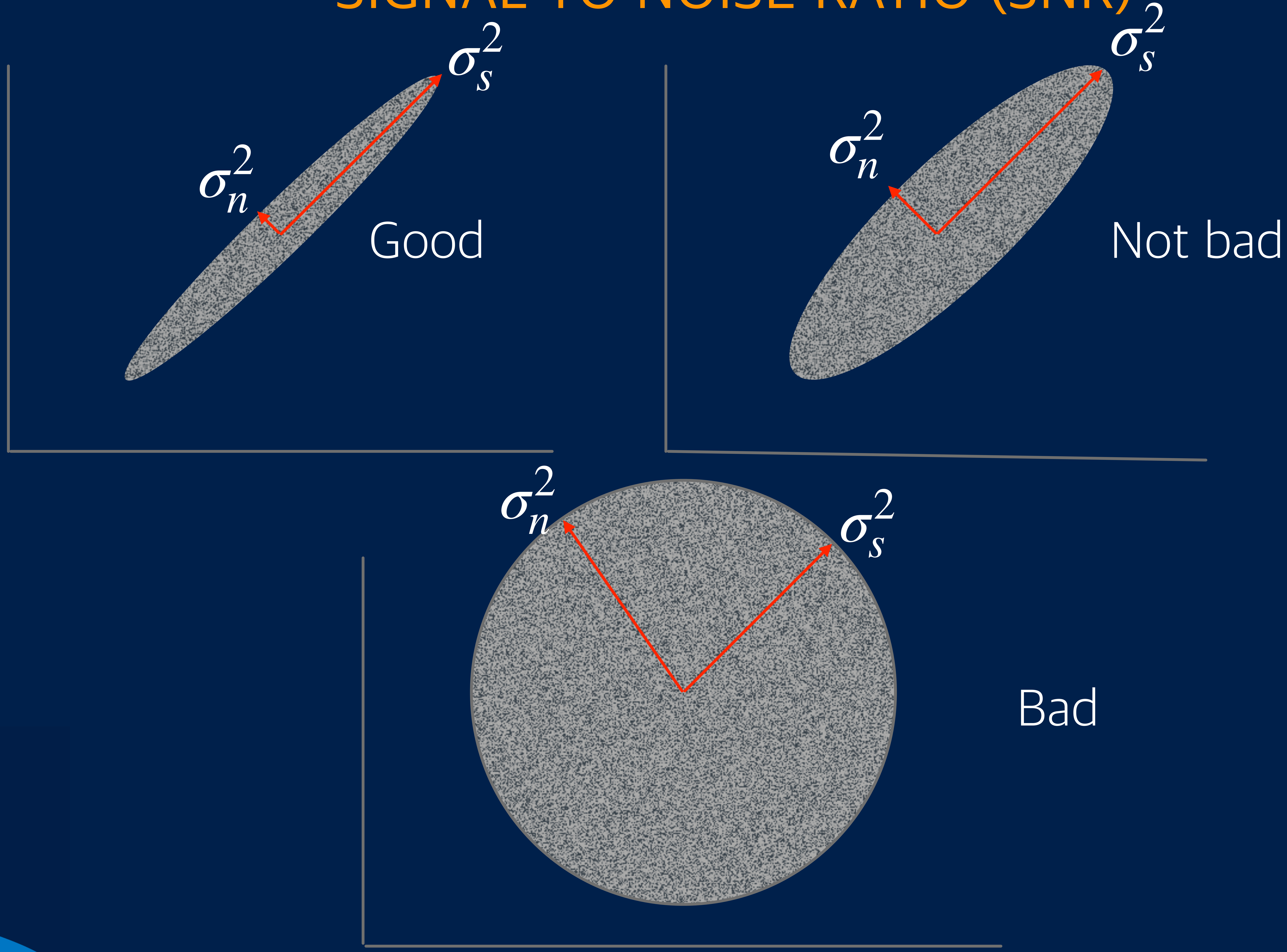
Subsituting $P^T = Q,$ we get

$$= \frac{1}{n-1} \Lambda$$

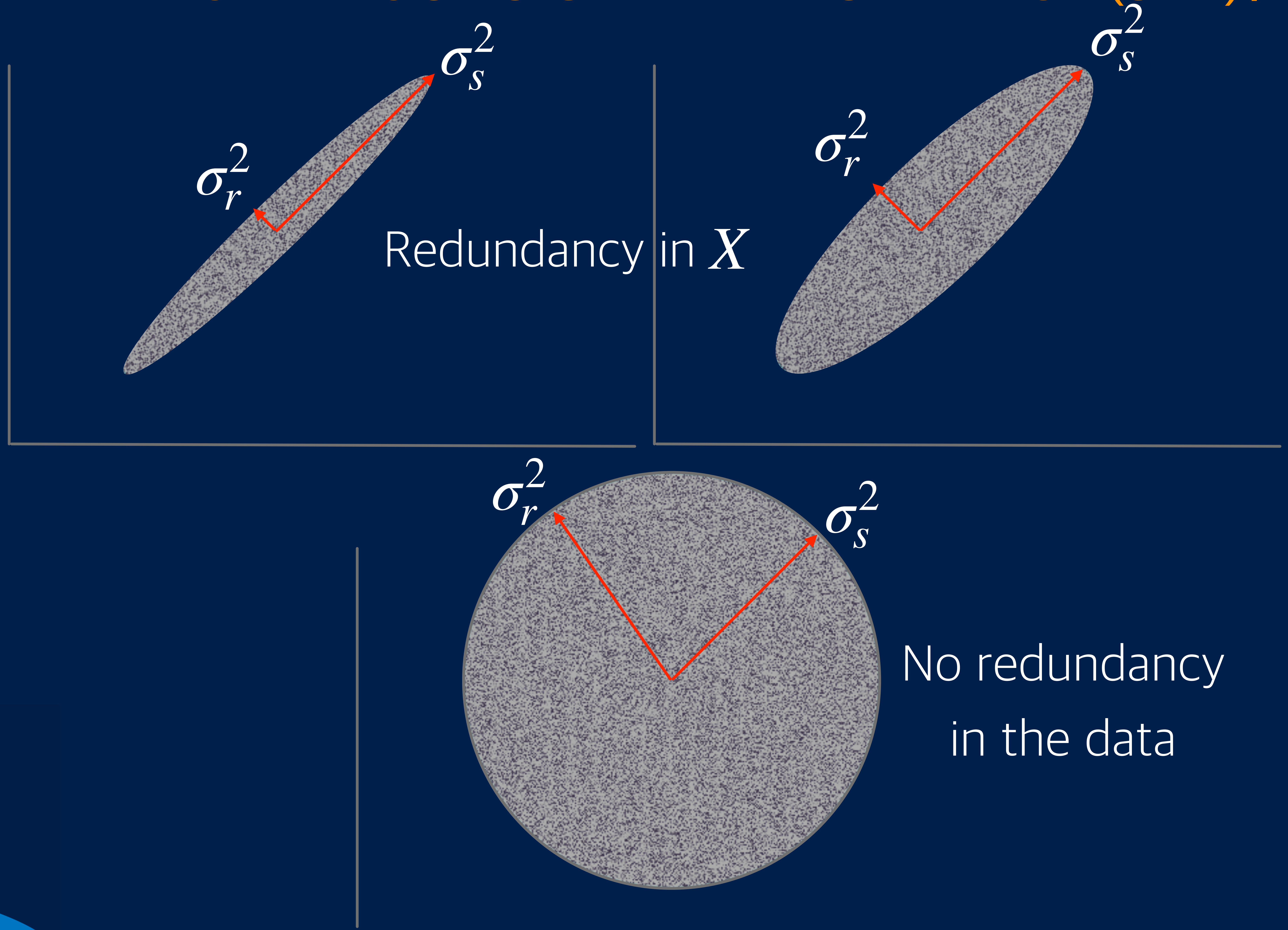
WORD VECTORS USING PCA

1. *Construct the co-occurrence matrix of size* $\boldsymbol{m} \times \boldsymbol{n}$ *matrix, where* $\boldsymbol{m}$ *is the vocabulary of the corpus and* $\boldsymbol{n}$ *is the number of context words*
2. *Subtract the mean from each row* $\boldsymbol{x_i}$
3. *Compute SVD to find eigen vectors and eigen values* ($\boldsymbol{A = U\Sigma V^T}$), *where* $\boldsymbol{A = XX^T}$
4. *The principal components of* $\boldsymbol{X}$ *are the eigenvectors of* $\boldsymbol{A}$
5. *The largest eigen value correspond to the principal component with a maximum variance*
6. *Find a suitable* $\boldsymbol{r}$ *to reduce the rank of* $\boldsymbol{U}$
7. *The rows of the eigen vectors* ($\boldsymbol{u_i}$) *correspond to the word vector of* $\boldsymbol{x_i}$

SIGNAL TO NOISE RATIO (SNR)



HOW ABOUT SIGNAL2REDUNDANCY (SRR)?



REDUNDANCY IN THE DATA

- Semantic redundancy
- Dimensional redundancy
  - Few basis vectors – many vectors are the linear combinations of the basis vectors
  - projecting onto a lower dimension makes sense
  - Orthogonal projection of the data onto a lower dimensional space

INTERPRETATION INTUITION

- Original matrix is decomposed into linearly independent components
- Each of the original component in broken into small number of components
- These components represent may be thought of representing the latent features of concepts present in the vocabulary
- The first component accounts for most of the possible variability of the original data
- The elements of the word embedding represents the strength of association with every underlying concepts
    - $\implies$ the meaning of every word is expressed as k components
    - Each latent feature column of U can be seen as a conceptual structure that characterizes a set of related terms
        - For example – Size, finance, fruit, color, $\cdots$ can be considered as hidden/latent abstract concepts
        - Words that fall into any of these concepts have similar word vectors

LATENT FACTOR ANALYSIS

Express the word vectors as a linear combination of latent factors

LATENT FACTOR MODELS

- Latent factor models study a random vector $X \in R^m$ by assuming that it is generated by a linear combination of a set of basis vectors
- $x = B\Lambda + \epsilon = B_1\lambda_1 + \cdots + B_K\lambda_K + \epsilon$
- where $B = [B_1, \cdots, , B_K]$ is the set of fixed but unknown basis. ϵ describes noise and s is the latent factor
- We want to minimize $\|X - B\Lambda\|_{F'}^2$ where $\|\cdot\|$ is the matrix Frobenius norm

$$\|A\|_F^2 = \sqrt{\sum_i^m \sum_j^n |A_{ij}|^2}$$

- Loadings represent degree to which each of the variables "correlates" with each of the latent factors
- Latent factor loadings reveals extent to which each of the variables contributes to the meaning of each of the factors.

LATENT FACTOR MODELS

$$X = B\Lambda + \Delta$$

$$\begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} \lambda_{11} & \cdots & \cdots & \lambda_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{n1} & \cdots & \cdots & \lambda_{nm} \end{bmatrix}_{n\times m} \begin{bmatrix} b_{11} \\ \cdots \\ \cdots \\ b_m \end{bmatrix}_{m\times 1} + \begin{bmatrix} \delta_1 \\ \cdots \\ \cdots \\ \delta_n \end{bmatrix}_{n\times 1}$$

WORD VECTORS COMPUTATION AS OPTIMIZATION PROBLEM?

- The goal is to learn word vectors that capture semantic relationships between words by minimizing an objective function
- Is this possible?
  - $\langle v_i, v_j \rangle \approx T(x_{ij})$
  - OR

$$min_{v,v'} \sum_{v,v'} (\langle v, v' \rangle - T(x_{ij})^2$$

$$\downarrow$$

  - $$min_{v,v'} \sum_{v,v'} f(\cdot)\Big(\langle v, v' \rangle - T(x_{ij})\Big)^2$$

OBJECTIVE FUNCTION AS LOG LIKELIHOOD

- Given a word w_n and its context w_{context}, the objective is to maximize the log-probability of the context words:
- $$\mathcal{L} = \sum_{n=1}^{|V|} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{n+j} \mid w_n)$$
    - The probability of observing the context word w_{n+j} given the target word w_n .

LATENT FACTORS

$\overrightarrow{apple}$		$\overrightarrow{apple} - \overrightarrow{iphone}$		$\overrightarrow{apple} - \overrightarrow{fruit}$	
iphone	0.266	raisin	0.574	ipad	0.412
ipad	0.287	pecan	0.576	iphone	0.433
apples	0.356	cranberry	0.584	macbook	0.435
blackberry	0.361	butternut	0.588	ipod	0.445
ipod	0.365	cider	0.591	imac	0.465
macbook	0.383	apricot	0.603	3gs	0.473
mac	0.391	tomato	0.607	Ipad	0.490
android	0.391	rosemary	0.615	itouch	0.512
google	0.395	rhubarb	0.615	ipad2	0.514
microsoft	0.418	feta	0.618	Iphone	0.514
ios	0.433	apples	0.623	ios	0.520
iphones	0.445	avocado	0.624	Macbook	0.524
touch	0.446	fennel	0.631	ibook	0.534
sony	0.447	chutney	0.631	IPhone	0.541

$$w_{apple} \approx \lambda_1 v_{company} + \lambda_2 v_{tech} + \lambda_3 v_{product} + \cdots + \lambda_n v_{fruit}$$

$$\langle w_i, \tilde{w}_k \rangle \approx \log(X_{ik})$$

SIMILAR WORDS

$\overrightarrow{language}$		$\overrightarrow{programming}$		$\overrightarrow{language} - \overrightarrow{programming}$		$\overrightarrow{language} - \overrightarrow{english}$	
languages	0.18	programing	0.29	language	0.55	constructs	0.62
English	0.25	programmers	0.38	english	0.57	domain-specific	0.63
spoken	0.30	programs	0.38	spoken	0.60	implicit	0.64
translation	0.34	programmer	0.39	pronunciation	0.60	object-oriented	0.64
vocabulary	0.34	program	0.40	malay	0.61	language	0.64
word	0.36	c++	0.40	spanish	0.61	scripting	0.65
learning	0.36	language	0.40	portuguese	0.61	concurrency	0.65
speaking	0.36	languages	0.42	hebrew	0.63	pervasive	0.66
grammar	0.36	java	0.42	etymology	0.63	semantics	0.66
meaning	0.37	visual	0.42	french	0.64	programming	0.66
linguistic	0.37	coding	0.43	afrikaans	0.64	cognition	0.66
words	0.38	development	0.43	nationality	0.65	declarative	0.67
spanish	0.38	learning	0.44	sanskrit	0.65	categorization	0.67
speak	0.39	applications	0.44	fluently	0.65	relational	0.67

$$w_{language} \approx \lambda_1 v_{NL} + \lambda_2 v_{CL} + \lambda_3 v_{grammar} + \cdots + \lambda_n v_{Tech}$$

ANOTHER EXAMPLE

$\overrightarrow{virus}$		$\overrightarrow{malware}$		$\overrightarrow{virus} - \overrightarrow{malware}$	
viruses	0.182	spyware	0.184	hepatitis	0.600
malware	0.303	viruses	0.246	herpes	0.600
infected	0.307	adware	0.264	vaccine	0.608
infection	0.315	virus	0.303	virus	0.611
spyware	0.344	malicious	0.304	encephalitis	0.619
viral	0.352	anti-virus	0.321	influenza	0.623
influenza	0.358	trojan	0.331	immunodeficiency	0.624
flu	0.375	phishing	0.343	fever	0.625
trojan	0.392	antivirus	0.379	disease	0.639
hiv	0.401	rootkits	0.386	dengue	0.642
vaccine	0.405	hackers	0.407	lymphoma	0.652
h1n1	0.405	rootkit	0.410	flu	0.654
hepatitis	0.414	spam	0.417	infectious	0.655

$$w_{virus} \approx \lambda_1 v_{virus_1} + \lambda_2 v_{virus_2} + \cdots + \lambda_n v_{virus_n}$$