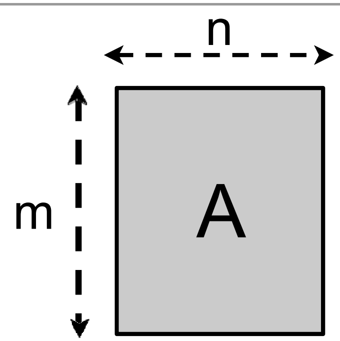
Computational Data mining

Fatemeh Mansoori

These slides are bases on the slides for course CS 363D by Prof. Pradeep Ravikumar anf for
course CX4242 by Mahdi Roozbahani

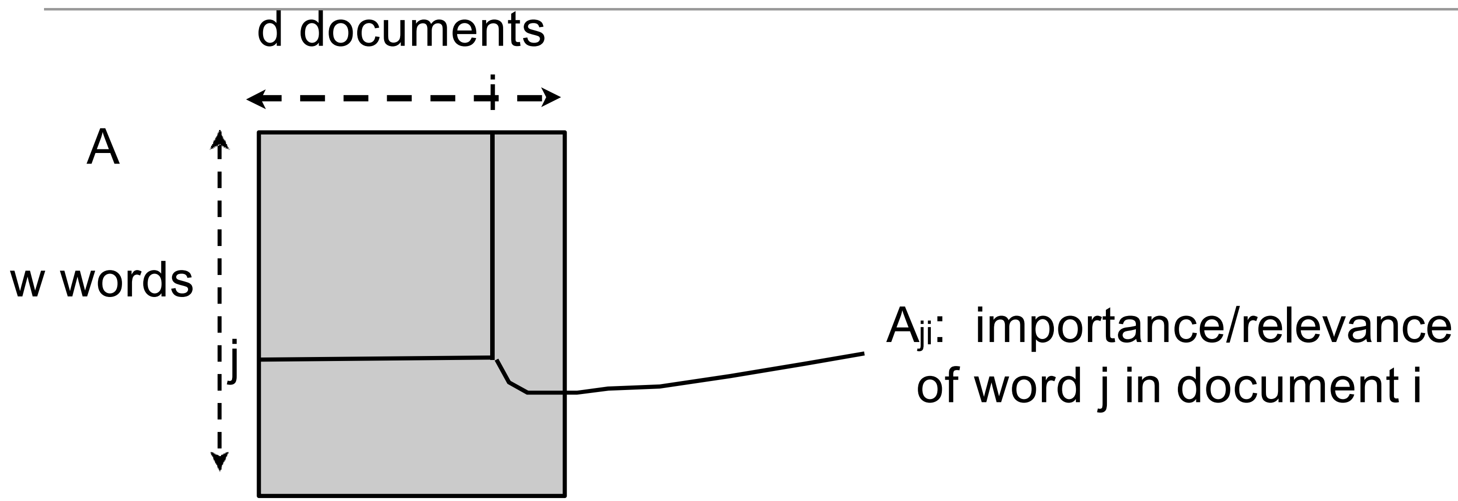
Matrices



- A matrix $A \in \mathbb{R}^{m \times n}$ can also be viewed as a **linear transformation**:

$$\begin{aligned} A : \mathbb{R}^n &\mapsto \mathbb{R}^m \\ x &\mapsto Ax \\ \alpha x + \beta y &\mapsto A(\alpha x + \beta y) = \alpha Ax + \beta Ay \quad : \textbf{Linear Transformation} \end{aligned}$$

Vector Space Model for Documents

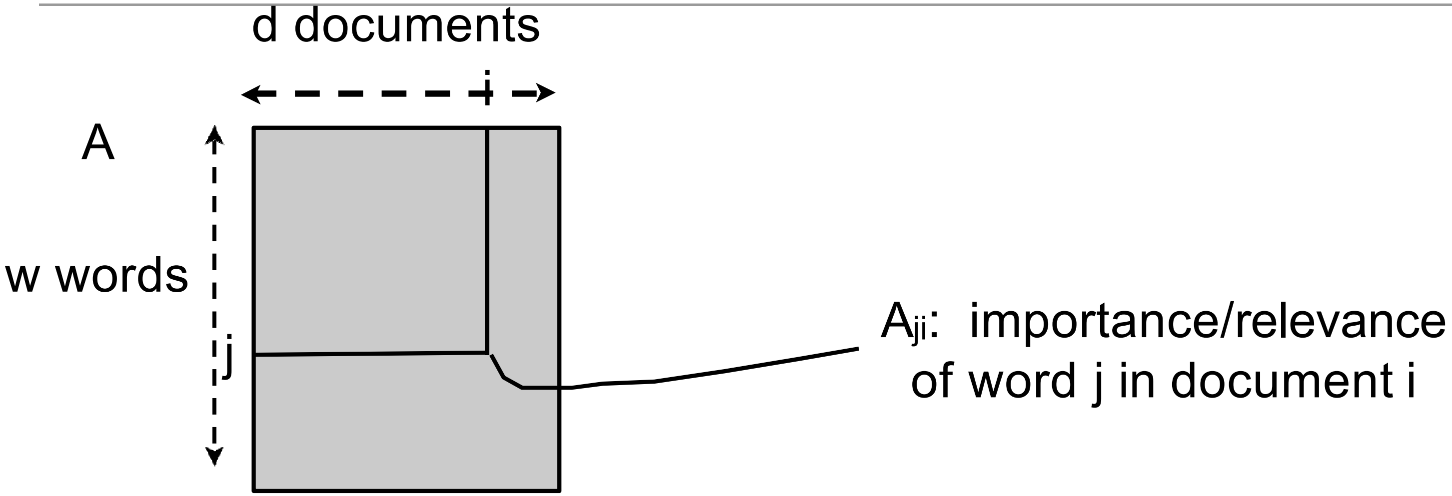


- Extract all unique words, ignoring case
- Eliminate **stop-words** : “a” , “and” , “the” , ...
- Eliminate non-content-bearing high-frequency and low-frequency words (using heuristic criteria)
- ….
- For each document, count no. of occurrences of each word

Vector Space Model for Documents

- Extract all unique words, ignoring case
- Eliminate **stop-words** : “a” , “and” , “the” , ...
- Eliminate non-content-bearing high-frequency and low-frequency words (using heuristic criteria)
- Extract word phrases (“New York”)
- Reduce words to their root/stem (eliminating plurals, tenses, pre/suffixes)
- Assign a unique integer between 1 and w to remaining w words
- For each document, count no. of occurrences of each word

Vector Space Model for Documents



- $A_{ji} = t_{ji} \times g_j \times s_i$

  t_{ji}: doc-term frequency; no. of times word j in document i

  g_j: importance of word i in entire document collection;
  e.g. $\log \frac{d}{d_j}$, where d_j is no. of documents that contains word j

  $s_i = 1/\sqrt{\sum_{j=1}^{w} (t_{ji} g_j)^2}$: normalization for document i.

- Note that columns of A are normalized: $\|a_i\|_2 = 1$.

TF-IDF

A word's importance score in a **document**, among **N documents**

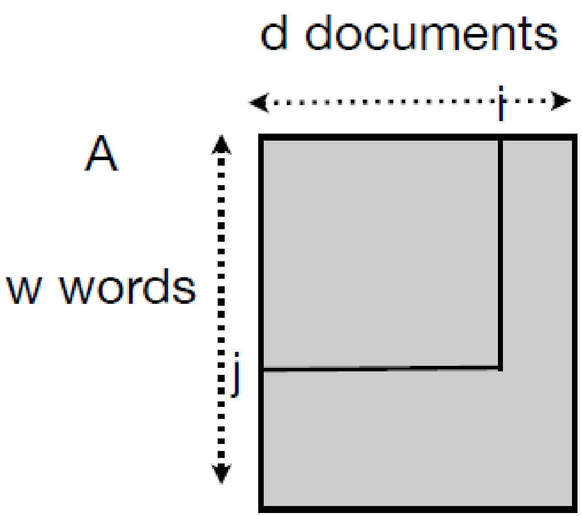
When to use it? Everywhere you use "word count", you can likely use TF-IDF.

TF: **term** frequency
= #appearance a **document**
(high, if terms appear many times in this document)

IDF: inverse document frequency
= log(**N** / #document containing that **term**)
(penalize "common" words appearing in almost any documents)

Final score = TF * IDF
(higher score → more "characteristic")

Query

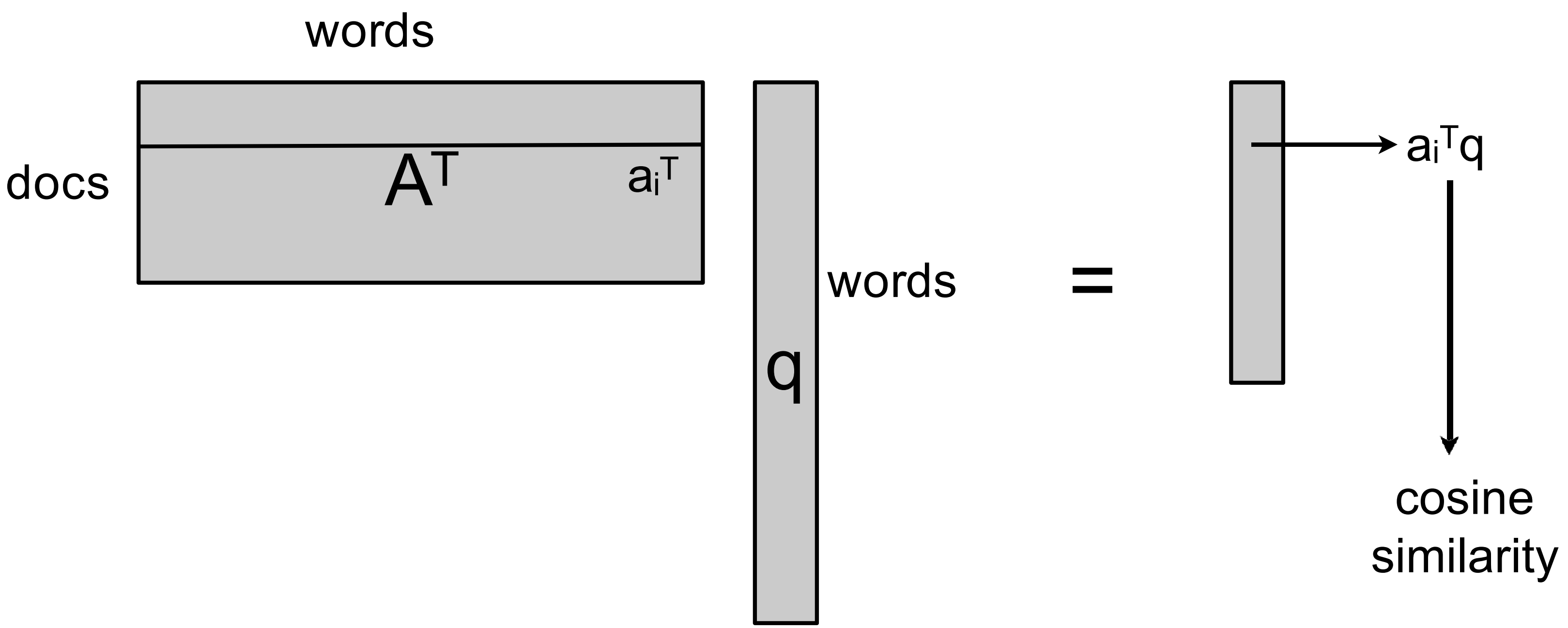


Suppose we query
"data mining"

$$q = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 0 \end{bmatrix}\begin{array}{l} \\ \\ w \\ \leftarrow \text{data} \\ \\ \leftarrow \text{mining} \\ \\ \\ \end{array}$$

$A^T q$: Scores of documents with respect to query

Query Retrieval



$$a_i^T q = \|a_i\|_2 \|q\|_2 \cos\theta$$

$A^T q$: Scores of documents with respect to query

Caveats with using word-document matrix

- **Size:** Even after pruning and following pre-processing steps outlined earlier, the number of words would be in the tens of thousands

- **Word Senses:**

  - Synonymy: different words have similar meaning
    e.g. searching for MRI, or “Magnetic Resonance Imaging”

  - Polysemy: One words may have different meanings depending on context
    e.g. “mining” could refer to “data mining” or “coal mining”

Caveats with using word-document matrix

- Imagine that we could convert word-document matrix, into an ideal "semantic term" - document matrix

- Imagine that given a query (which like a document is a set of words), we can convert it into a set of "semantic terms"

  - Then we could compute query-document similarities as before

  - We humans do this all the time

  - Think of this ideal "semantic term" - document matrix as "approximating" our word-document matrix

How Good is my Matrix Approximation?

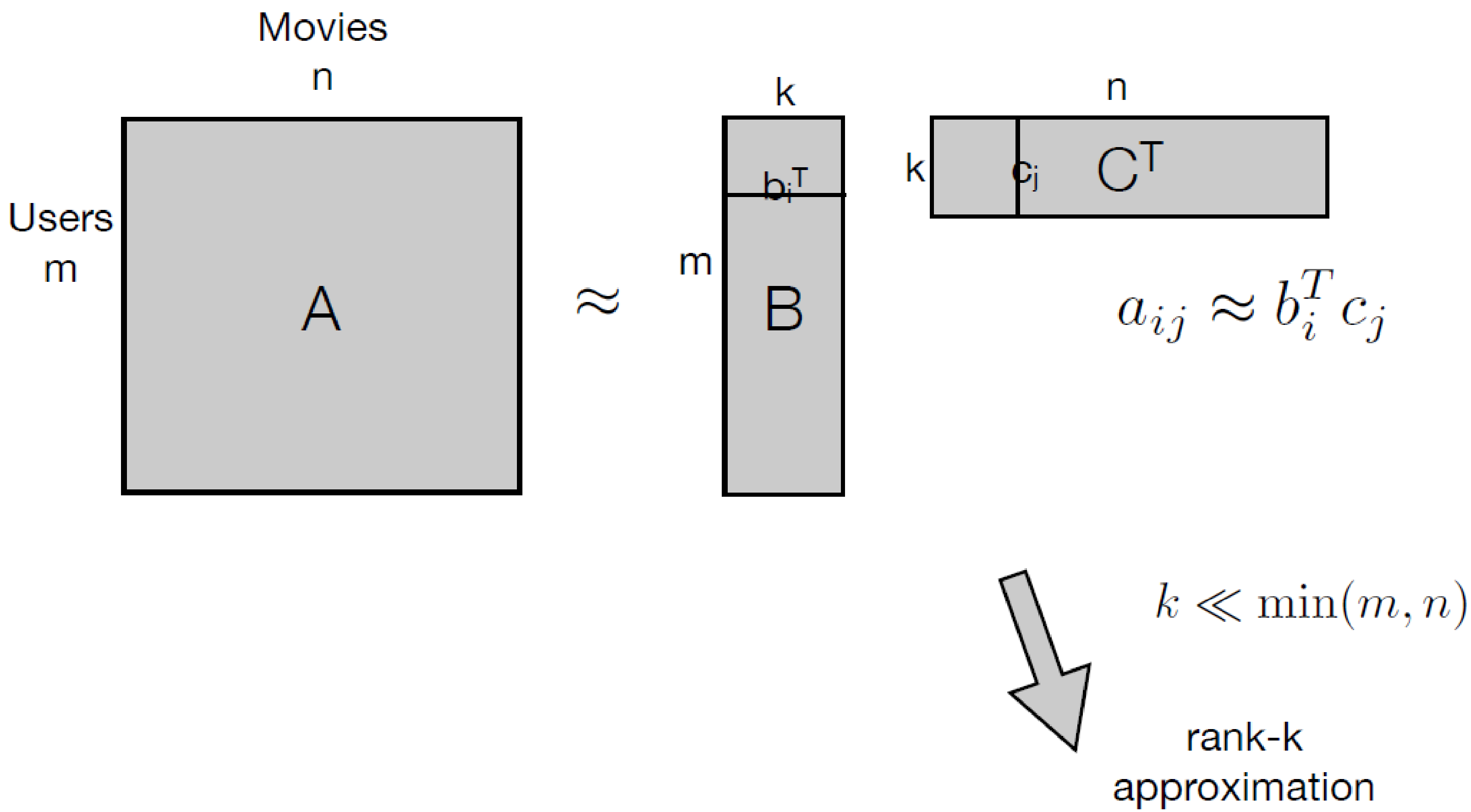
- Bill Gates, Lord Kelvin: You can’t really make progress unless you can **measure**!

- Suppose I want to **approximate** a matrix A by another matrix B.
  - How good is B as an approximation?
  - Matrices also have norms || A ||
  - Use a matrix norm to measure approximation error: || A-B ||
  - A popular matrix norm is the Frobenius norm:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

How do I Approximate my Matrix?

- A popular way to approximate a big matrix: Low-rank Approximation



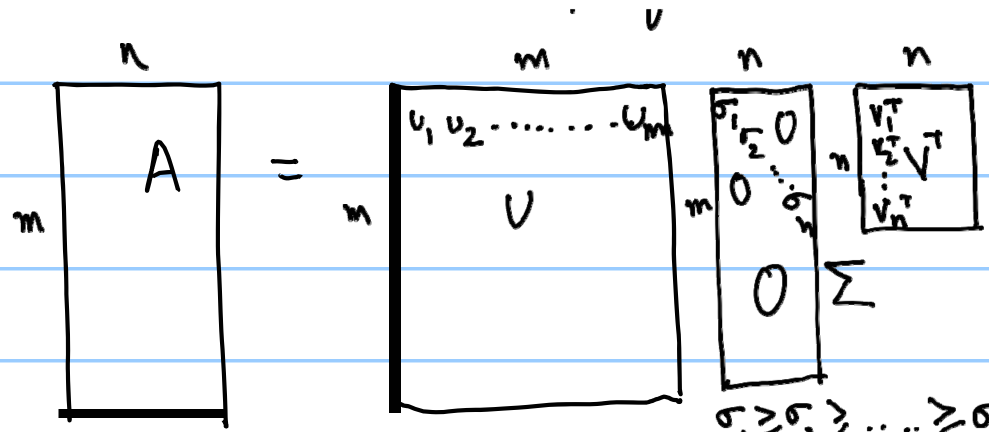
Best Low-rank Approximation

- Given A, and k, what is the best rank-k approximation?
- Find matrices B, C **of rank-**k which solve:

$$\min_{B,C} \|A - B\,C^T\|_F.$$

- Solution is given by SVD: Singular Value Decomposition of A

SVD; Singular Value Decomposition



$$\left(A = U\Sigma V^T\right)$$

$$\underbrace{\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0}_{\text{singular values}}$$

orthogonal matrix

$$U^T U = I$$

$$u_i^T u_j = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$$

left singular vectors

orthogonal matrix

$$V^T V = I$$

$$v_i^T v_j = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$$

right singular vectors

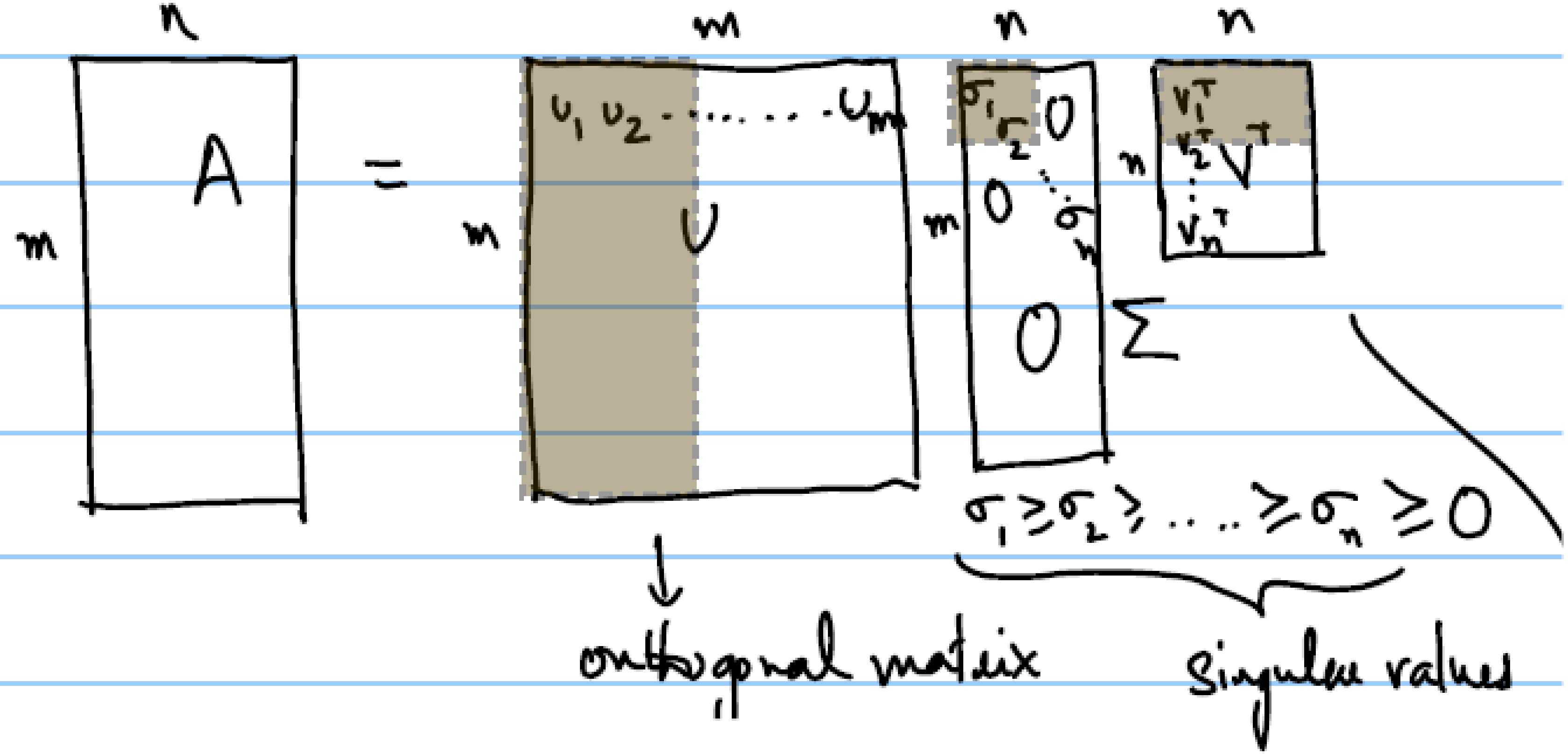
Low-Rank Approximation Using SVD

$$
\begin{aligned}
\mathbf{A}_k &= \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T \\
&= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}
\end{aligned}
$$

where $U_k^T U_k = I$, $V_k^T V_k = I$, and

$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$.

Low-Rank Approximation Using SVD



Low-Rank Approximation Using SVD

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

$$= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}$$

Important Result: Among all rank-k approximations of A, the best is A_k:

$\min_{B:\,\mathrm{rank}(B)\leq k} \|A - B\|_F \qquad \leftarrow$ minimizing $B = A_k$.

Latent Semantic Indexing (LSI)

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

$$= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}$$

- Use A_k instead of A for computing query-document similarities.
- Use $A_k^T\, q$ instead of $A^T\, q$.

LSI Contd.

Note that $\mathbf{A}_k^T\mathbf{q} = (\mathbf{V}_k\mathbf{\Sigma}_k)\left(\mathbf{U}_k^T\mathbf{q}\right)$.

$$\mathbf{U}_k^T\mathbf{q} \;=\; \begin{bmatrix} \mathbf{u}_1^T\mathbf{q} \\ \mathbf{u}_2^T\mathbf{q} \\ \vdots \\ \mathbf{u}_k^T\mathbf{q} \end{bmatrix},$$

- Each component $u_i^T q$ of the vector $U_k^T q$ is the **projection** of query vector q onto the singular vector u_i.
- The w-dimensional query vector q is reduced to k dimensions
- The singular vectors (u_i's) do not span all possible documents, but span "important" part of the space

LSI Contd.

Note that $\mathbf{A}_k^T \mathbf{q} = (\mathbf{V}_k \mathbf{\Sigma}_k) \left(\mathbf{U}_k^T \mathbf{q}\right)$.

$$A = U_k \Sigma_k V_k^T + U_{w-k} \Sigma_{w-k} V_{w-k}^T$$

$$U_k^T A = \Sigma_k V_k^T \qquad \ldots\ U \text{ is orthonormal}$$

- Thus, $V_k \Sigma_k$ is the projection of the documents (columns of A) onto U_k.
- So that $A_k^T q = (V_k \Sigma_k)(U_k^T q)$ can be interpreted as dot-product between projected documents and projected query!

LSI: Alternatively

1. For the entire document collection, form $\mathbf{V}_k\mathbf{\Sigma}_k$.

2. For a new query $\mathbf{q}$, form $\mathbf{U}_k^T\mathbf{q}$.

3. Compute $\mathbf{z} = (\mathbf{V}_k\mathbf{\Sigma}_k)(\mathbf{U}_k^T\mathbf{q})$ and return the document i with large $\mathbf{z}_i$ values as being the most relevant.

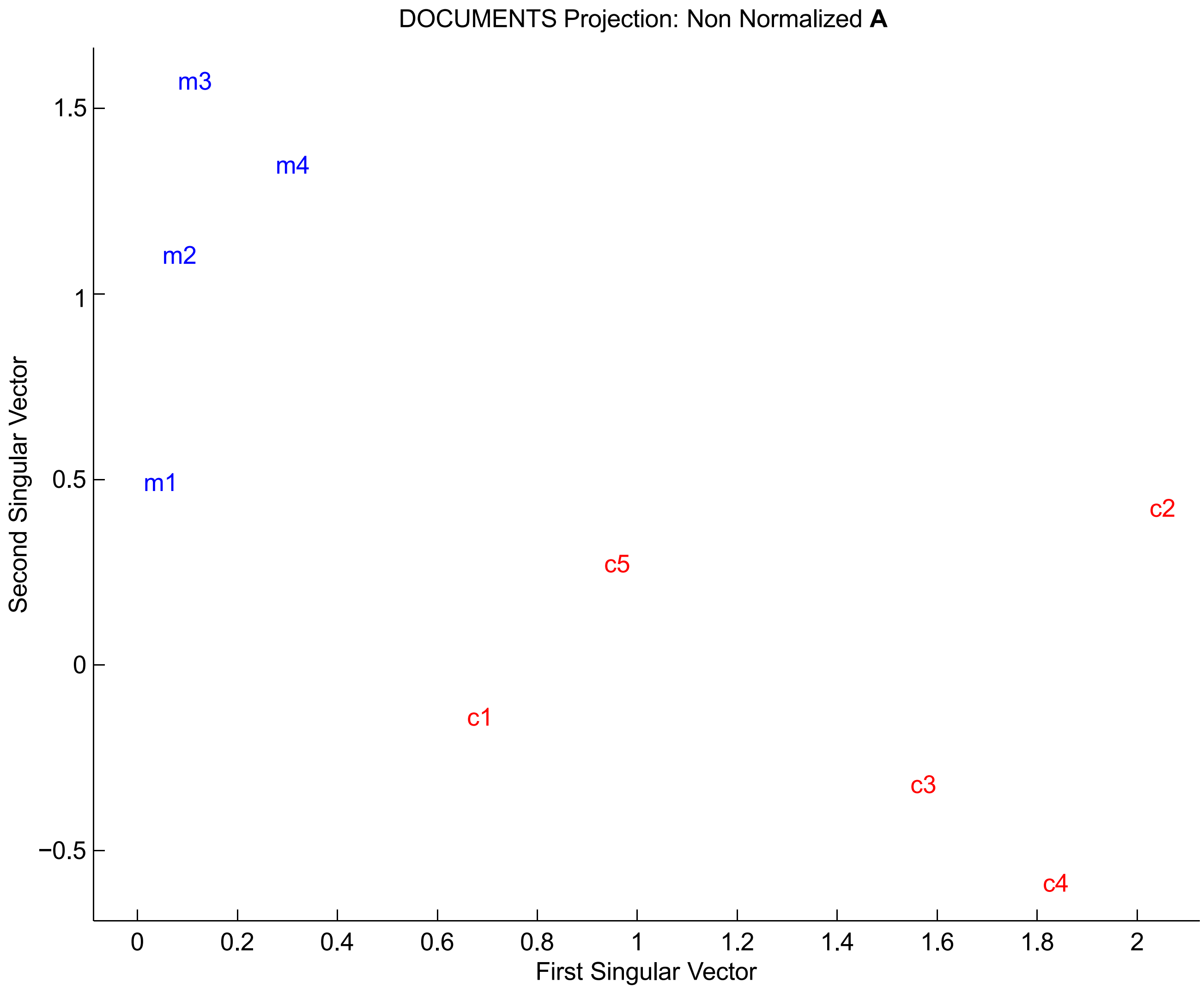
Term-Doc. Matrix (Vector Space Model)

Documents

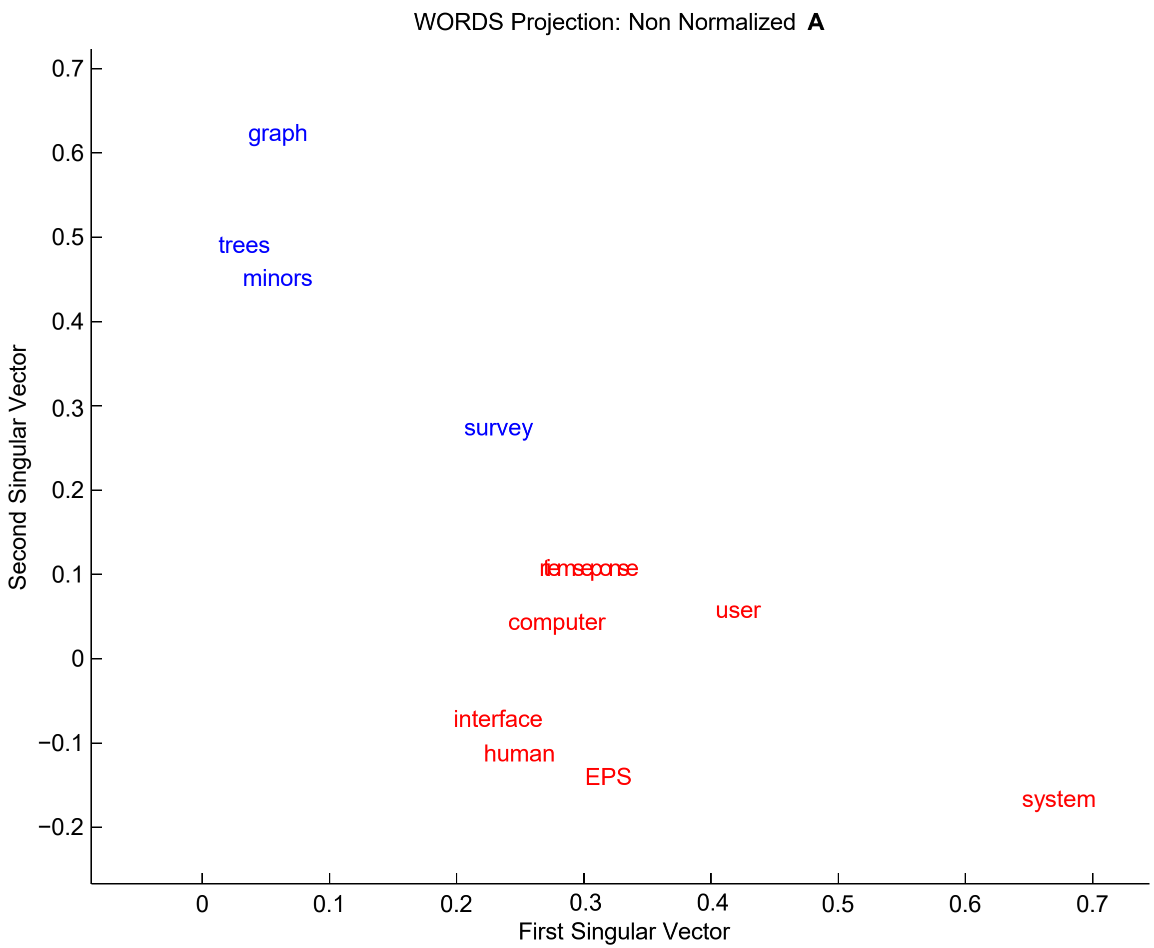
Terms	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

A: Nine document vectors, each in a 12 dimensional word space

Documents projected onto 1st two sing. vectors



Words projected onto same two right sing. vectors



Words projected on 2-D space from **A**

An other look in LSI

Latent Semantic Indexing (LSI)

We want to find the $A^T q$

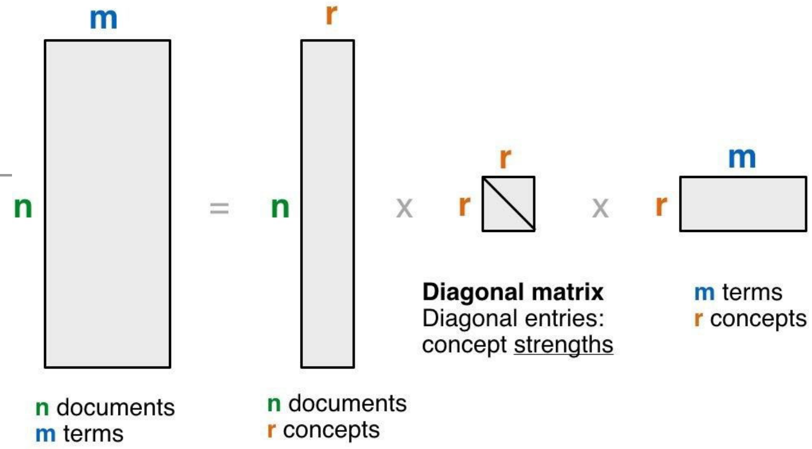
A is term-document matrix so A^T is document-term matrix

document-term matrix

	data	system	retireval	lung	ear
doc1	1	1	1		
doc2	1	1	1		
doc3				1	1
doc4				1	1

SVD - Example

$$A = U \quad \Lambda \quad V^T$$



$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{data} & \text{info} & \text{retrieval} & \text{brain} & \text{lung} \end{array} \\
\begin{array}{c}\text{CS docs}\\ \\ \\ \\ \text{MD docs}\end{array} & \begin{bmatrix}1&1&1&0&0\\2&2&2&0&0\\1&1&1&0&0\\5&5&5&0&0\\0&0&0&2&2\\0&0&0&3&3\\0&0&0&1&1\end{bmatrix}
\end{array}
=
\begin{array}{c} \begin{array}{cc}\text{CS concept} & \text{MD concept}\end{array}\\ \begin{bmatrix}0.18&0\\0.36&0\\0.18&0\\0.90&0\\0&0.53\\0&0.80\\0&0.27\end{bmatrix}\end{array}
\times
\begin{bmatrix}9.64&0\\0&5.29\end{bmatrix}
\times
\begin{array}{cc} & \begin{array}{ccccc}\text{data}&\text{info}&\text{retrieval}&\text{brain}&\text{lung}\end{array}\\ & \begin{bmatrix}0.58&0.58&0.58&0&0\\0&0&0&0.71&0.71\end{bmatrix} & \begin{array}{c}\text{CS concept}\\ \text{MD concept}\end{array}\end{array}
$$

“strength” of CS-concept

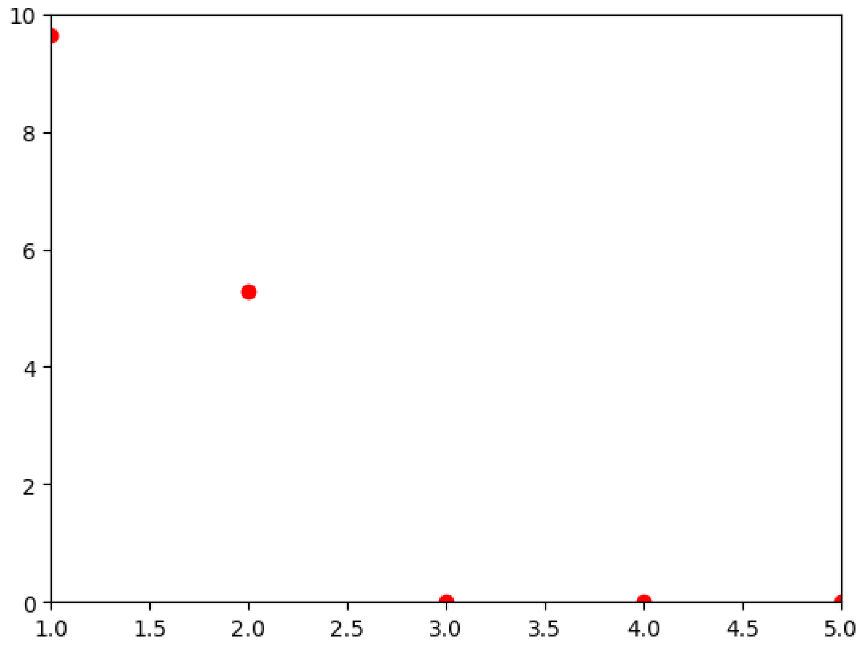
document-concept
similarity matrix

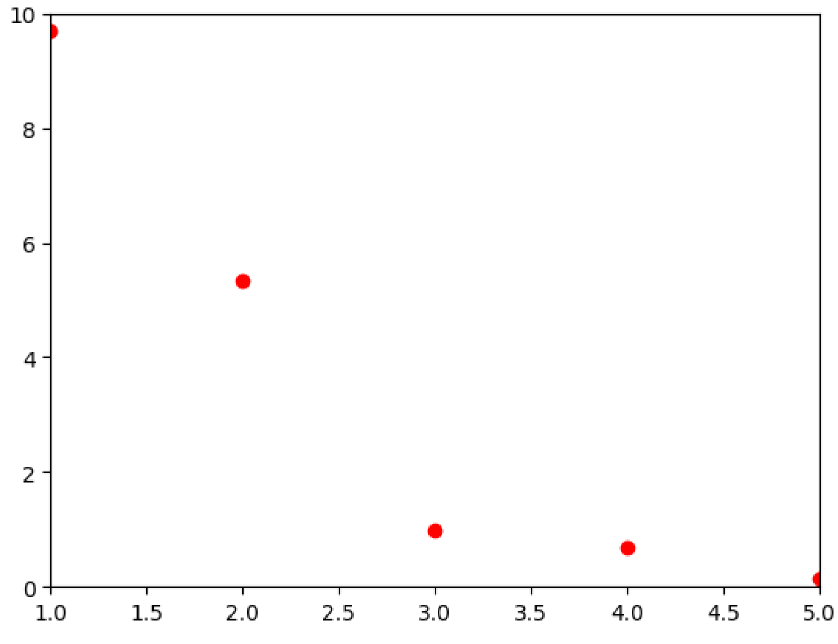
term-concept
similarity matrix

Singular values plot

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

1	1	1	0	0
2	2	2	0	0
1	1	1	0	1
5	5	5	0	0
0	1	0	2	2
0	0	1	3	3
0	0	0	1	1





determining the optimal number of dimensions

- actual number of dimensions that can be used is limited by the number of documents in the collection
- around 300 dimensions will usually provide the best results with moderate-sized document collections (hundreds of thousands of documents)
- 400 dimensions for larger document collections (millions of documents)
- When LSI topics are used as features in supervised learning methods, one can use prediction error measurements to find the ideal dimensionality.

SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:

U: document-concept similarity matrix

V: term-concept similarity matrix

Λ: diagonal elements: concept “strengths”

SVD - Interpretation #1

'documents', 'terms' and 'concepts':

Q**:** if **A** is the document-to-term matrix, what is the similarity matrix $\mathbf{A}^T \mathbf{A}$?

A:

Q: $\mathbf{A} \mathbf{A}^T$?

A:

SVD -Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & \cancel{5.29} \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

SVD -Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & \cancel{5.29}
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

SVD -Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Case Study

How to do queries with LSI?

Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?
A: map query vectors into ‘concept space’ – how?

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{data} & \text{info} & \text{retrieval} & \text{brain} & \text{lung} \end{array} \\
\begin{array}{c} \\ \text{CS docs} \\ \\ \\ \\ \text{MD docs} \\ \\ \end{array} &
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

Evaluation

Confusion Matrix

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Accuracy

$$\textbf{\textit{Accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Accuracy is useful when the target class is ***well balanced*** but is not a good choice for the unbalanced classes.

- scenario where we had 99 images of the dog and only 1 image of a cat present in our training data
  - our model would always predict the dog, and therefore we got 99% accuracy.

Precision and Recall

- Precision
  - defined as the number of true positives divided by the number of predicted positives.

- False Positive is a higher concern than False Negatives.

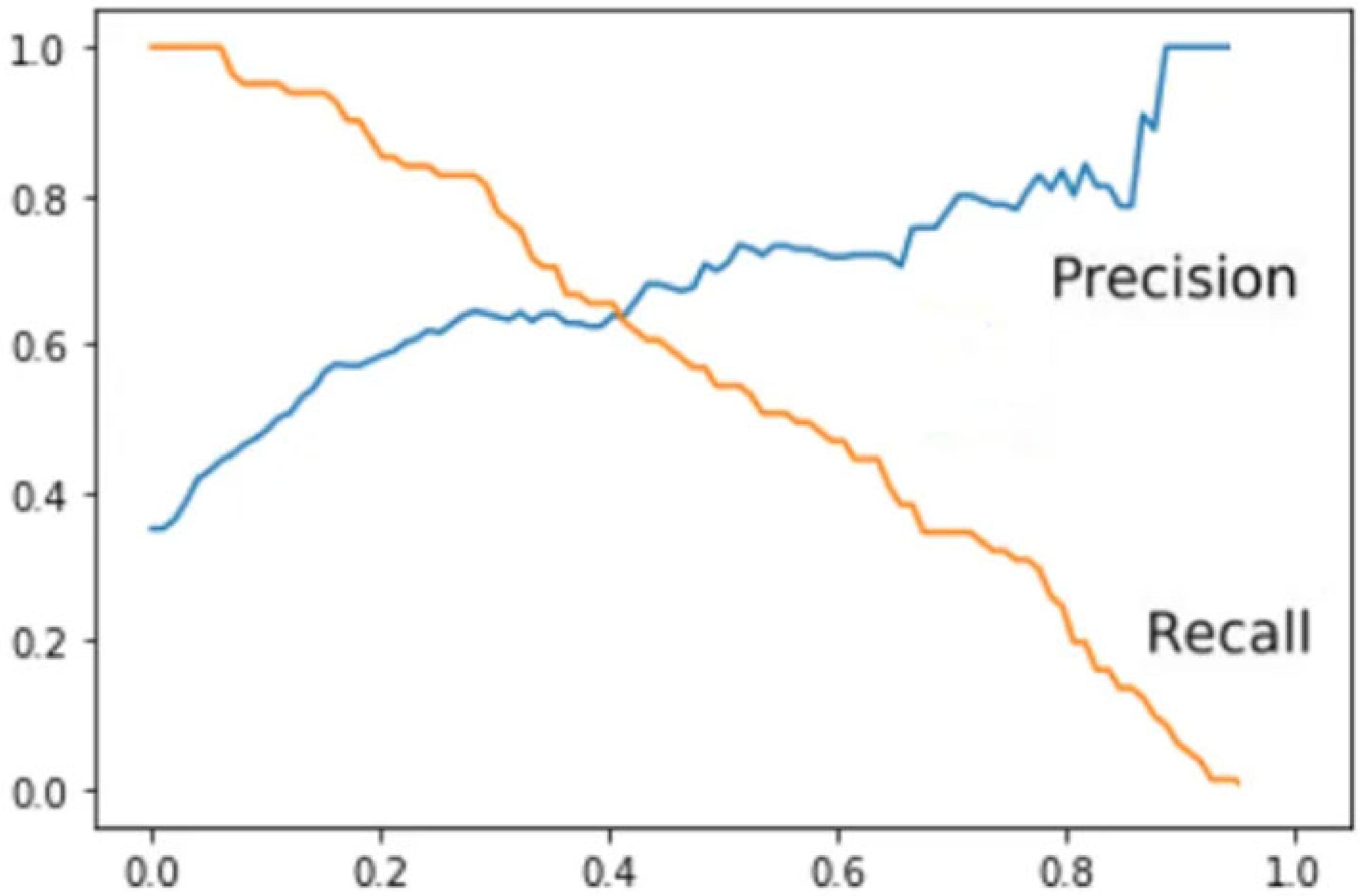
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- Recall (Sensitivity)
  - defined as the number of true positives divided by the total number of actual positives.

- False Negative is of higher concern than False Positive

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Precision/Recall Tradeoff



F measure

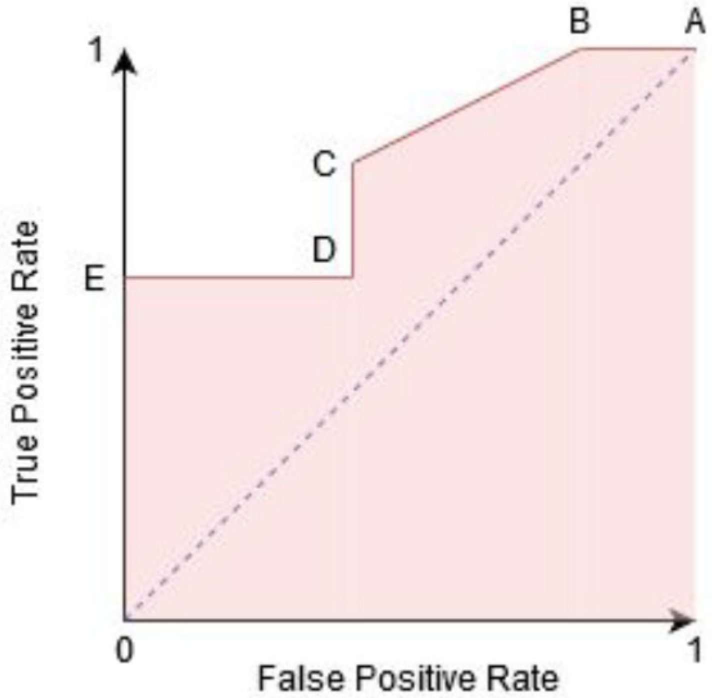
The F1 score is easily one of the most reliable ways to score how well a classification model performs. It is the weighted average of precision and recall

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{(\text{precision} + \text{recall})}$$

Receiver-Operator Curve (ROC Curve) and Area Under the Curve (AUC)

- plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values

- **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes.



Evaluation in information retrieval

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Recall@k

- Number of TP in the first k retrieved documents divided by the number of all positives
- K refers to the number of items returned by the IR system

Precision@k

- Number of TP in the first k retrieved documents divided by the k