COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2020.

Lecture 17

LOGISTICS

- Problem Set 3 deadline extended until Monday 10/26, 8pm.
- Week 9 Quiz will now be due Tuesday 10/27, 8pm.

Last Few Classes: Low-Rank Approximation and PCA

- Compress data that lies close to a k-dimensional subspace.
- Equivalent to finding a low-rank approximation of the data matrix $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$ for orthonormal $\mathbf{V} \in \mathbb{R}^{d \times k}$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).
- Singular vectors of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$. Singular values squared are the eigenvalues.

Last Few Classes: Low-Rank Approximation and PCA

- Compress data that lies close to a k-dimensional subspace.
- Equivalent to finding a low-rank approximation of the data matrix $\mathbf{X}$: $\mathbf{X} \approx \mathbf{X}\mathbf{V}\mathbf{V}^T$ for orthonormal $\mathbf{V} \in \mathbb{R}^{d \times k}$.
- Optimal solution via PCA (eigendecomposition of $\mathbf{X}^T\mathbf{X}$ or equivalently, SVD of $\mathbf{X}$).
- Singular vectors of $\mathbf{X}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$. Singular values squared are the eigenvalues.

This Class: Applications of low-rank approx. beyond compression.

- Matrix completion and collaborative filtering
- Entity embeddings (word embeddings, node embeddings, etc.)
- Low-rank approximation fornon-linear dimensionality reduction.
- Spectral graph theory, spectral clustering.

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix).

MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.

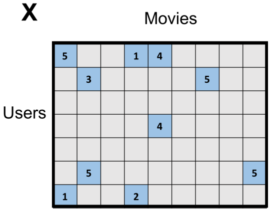
X

Movies

Users

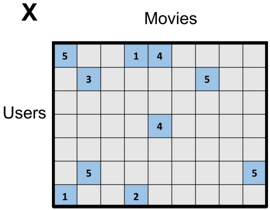
5	3	3	1	4	4	4	3	5
4	3	3	1	4	4	5	3	5
3	3	3	2	3	3	3	3	3
4	3	3	4	4	4	4	3	3
3	3	3	2	3	3	3	3	3
2	5	3	4	4	4	4	4	5
1	3	3	2	3	3	3	1	2

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



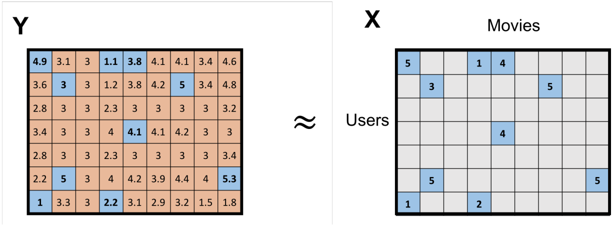
MATRIX COMPLETION

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



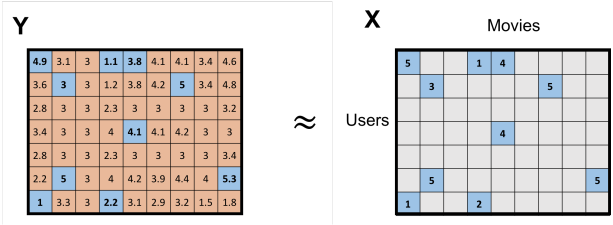
Solve: $Y = \underset{\text{rank }-k\ \mathbf{B}}{\arg\min} \sum_{\text{observed }(j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



Solve: $Y = \underset{\text{rank } -k \ \mathbf{B}}{\arg\min} \sum_{\text{observed } (j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.



Solve: $Y = \underset{\text{rank }-k \ \mathbf{B}}{\arg\min} \sum_{\text{observed } (j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

Under certain assumptions, can show that $\mathbf{Y}$ well approximates $\mathbf{X}$ on both the observed and (most importantly) unobserved entries.

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ which we cannot fully observe but believe is close to rank-k (i.e., well approximated by a rank k matrix). Classic example: the Netflix prize problem.

Y

4.9	3.1	3	**1.1**	**3.8**	4.1	4.1	3.4	4.6
3.6	**3**	3	1.2	3.8	4.2	**5**	3.4	4.8
2.8	3	3	2.3	3	3	3	3	3.2
3.4	3	3	4	**4.1**	4.1	4.2	3	3
2.8	3	3	2.3	3	3	3	3	3.4
2.2	**5**	3	4	4.2	3.9	4.4	4	**5.3**
1	3.3	3	**2.2**	3.1	2.9	3.2	1.5	1.8

$\approx$

X

Movies

Users

5	3	3	1	4	4	4	3	5
4	3	3	1	4	4	5	3	5
3	3	3	2	3	3	3	3	3
4	3	3	4	4	4	4	3	3
3	3	3	2	3	3	3	3	3
2	5	3	4	4	4	4	4	5
1	3	3	2	3	3	3	1	2

Solve: $Y = \underset{\text{rank}-k\ \mathbf{B}}{\arg\min} \sum_{\text{observed } (j,k)} \left[\mathbf{X}_{j,k} - \mathbf{B}_{j,k}\right]^2$

Under certain assumptions, can show that $\mathbf{Y}$ well approximates $\mathbf{X}$ on both the observed and (most importantly) unobserved entries.

Dimensionality reduction embeds d-dimensional vectors into k dimensions. But what about when you want to embed objects other than vectors?

Dimensionality reduction embeds d-dimensional vectors into k dimensions. But what about when you want to embed objects other than vectors?

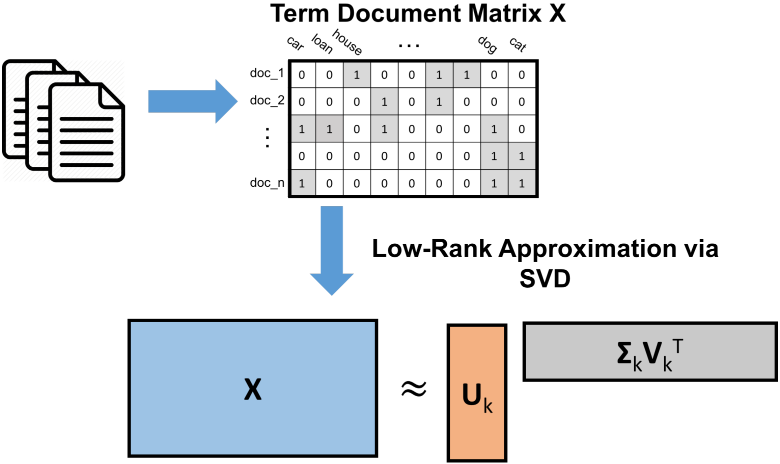
- Documents (for topic-based search and classification)
- Words (to identify synonyms, translations, etc.)
- Nodes in a social network

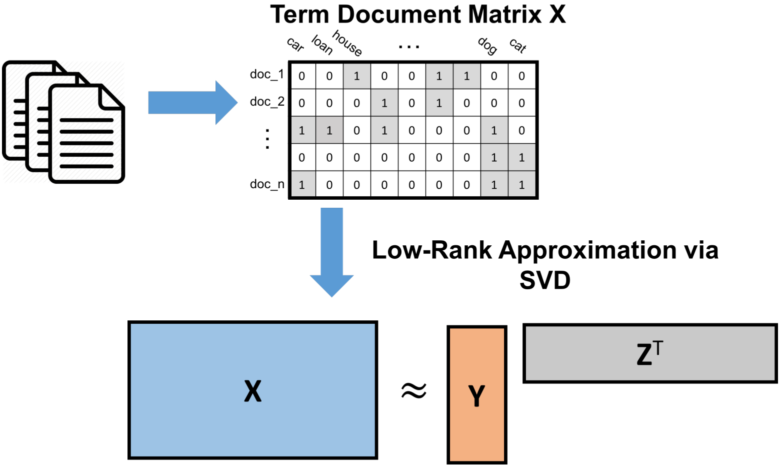
Dimensionality reduction embeds d-dimensional vectors into k dimensions. But what about when you want to embed objects other than vectors?

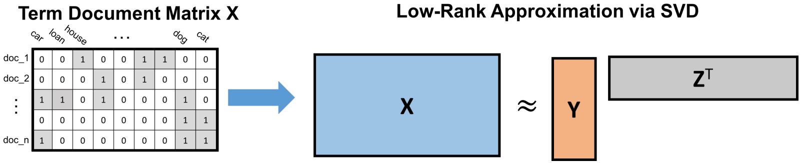
- Documents (for topic-based search and classification)
- Words (to identify synonyms, translations, etc.)
- Nodes in a social network

Usual Approach: Convert each item into a high-dimensional feature vector and then apply low-rank approximation.

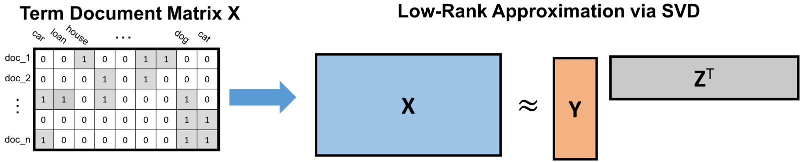
EXAMPLE: LATENT SEMANTIC ANALYSIS




Term Document Matrix X
car loan house . . . dog cat
doc_1 0 0 1 0 0 1 1 0 0
doc_2 0 0 0 1 0 1 0 0 0
⋮
1 1 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1 1
doc_n 1 0 0 0 0 0 0 1 1
Low-Rank Approximation via SVD
X
≈
Y
Z^T


Term Document Matrix X
car loan house ... dog cat
doc_1 0 0 1 0 0 1 1 0 0
doc_2 0 0 0 1 0 1 0 0 0
⋮ 1 1 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1 1
doc_n 1 0 0 0 0 0 0 1 1
Low-Rank Approximation via SVD
X ≈ Y Z^T

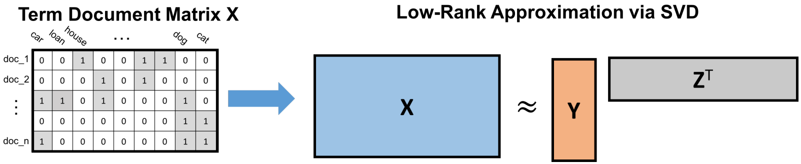
EXAMPLE: LATENT SEMANTIC ANALYSIS



- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

$$\mathbf{X}_{i,a} \approx (\mathbf{Y}\mathbf{Z}^T)_{i,a} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

EXAMPLE: LATENT SEMANTIC ANALYSIS

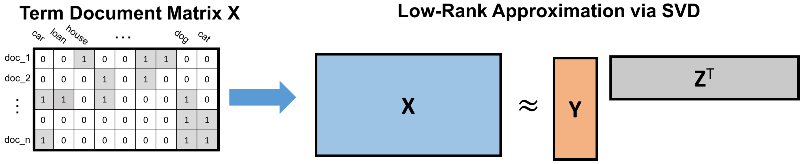


- If the error $\|\mathbf{X} - \mathbf{Y}\mathbf{Z}^T\|_F$ is small, then on average,

$$\mathbf{X}_{i,a} \approx (\mathbf{Y}\mathbf{Z}^T)_{i,a} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

- I.e., $\langle \vec{y}_i, \vec{z}_a \rangle \approx 1$ when doc_i contains $word_a$.

EXAMPLE: LATENT SEMANTIC ANALYSIS

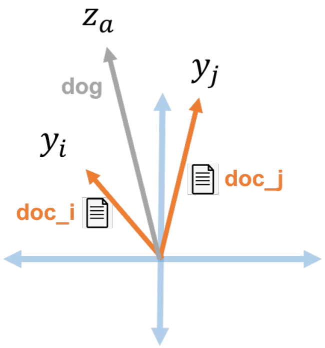


- If the error $\|X - YZ^T\|_F$ is small, then on average,

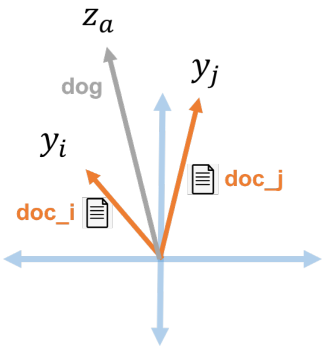
$$X_{i,a} \approx (YZ^T)_{i,a} = \langle \vec{y}_i, \vec{z}_a \rangle.$$

- I.e., $\langle \vec{y}_i, \vec{z}_a \rangle \approx 1$ when doc_i contains $word_a$.
- If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$.

If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$

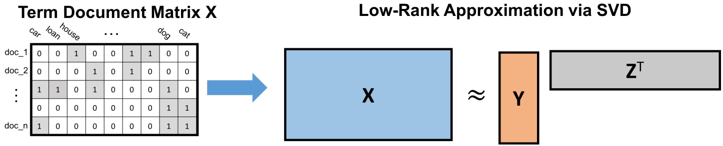


If doc_i and doc_j both contain $word_a$, $\langle \vec{y}_i, \vec{z}_a \rangle \approx \langle \vec{y}_j, \vec{z}_a \rangle \approx 1$



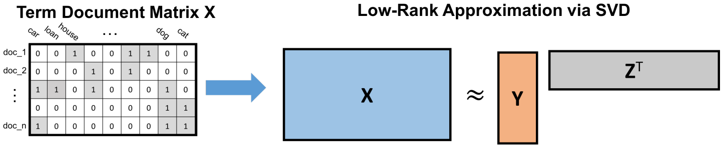
Another View: Each column of **Y** represents a 'topic'. $\vec{y}_i(j)$ indicates how much doc_i belongs to topic j. $\vec{z}_a(j)$ indicates how much $word_a$ associates with that topic.

EXAMPLE: LATENT SEMANTIC ANALYSIS



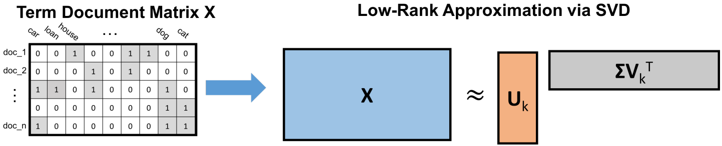
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.

EXAMPLE: LATENT SEMANTIC ANALYSIS



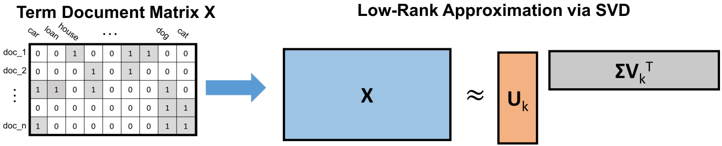
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$.

EXAMPLE: LATENT SEMANTIC ANALYSIS



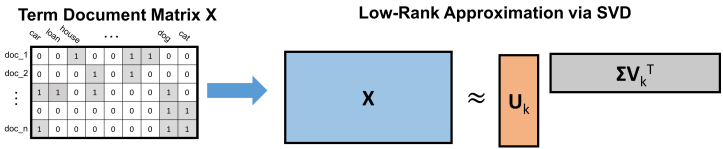
- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \boldsymbol{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\mathbf{X}^T\mathbf{X}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T$.

EXAMPLE: LATENT SEMANTIC ANALYSIS



- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\mathbf{X}^T\mathbf{X}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$.
- What is the best rank-k approximation of $\mathbf{X}^T\mathbf{X}$? I.e. $\arg\min_{\text{rank}-k\ \mathbf{B}} \|\mathbf{X}^T\mathbf{X} - \mathbf{B}\|_F$

EXAMPLE: LATENT SEMANTIC ANALYSIS



- Just like with documents, $\vec{z}_a$ and $\vec{z}_b$ will tend to have high dot product if $word_a$ and $word_b$ appear in many of the same documents.
- In an SVD decomposition we set $\mathbf{Z}^T = \mathbf{\Sigma}_k \mathbf{V}_K^T$.
- The columns of $\mathbf{V}_k$ are equivalently: the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$. The eigendecomposition of $\mathbf{X}^T\mathbf{X}$ is $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$.
- What is the best rank-k approximation of $\mathbf{X}^T\mathbf{X}$? I.e. $\arg\min_{\text{rank}-k \ \mathbf{B}} \|\mathbf{X}^T\mathbf{X} - \mathbf{B}\|_F$
- $\mathbf{X}^T\mathbf{X} = \mathbf{V}_k\mathbf{\Sigma}_k^2\mathbf{V}_k^T = \mathbf{Z}\mathbf{Z}^T$.

LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of $X^T X$: where $(X^T X)_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.

LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.

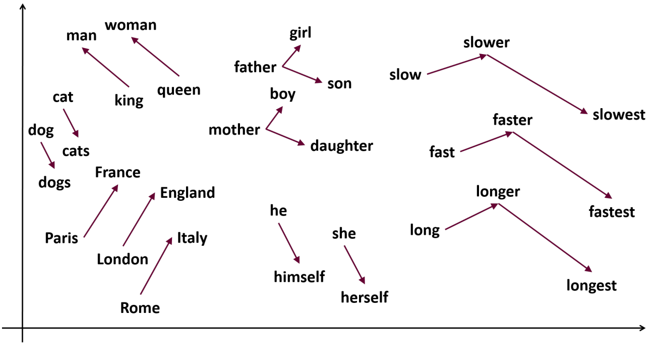
LSA gives a way of embedding words into k-dimensional space.

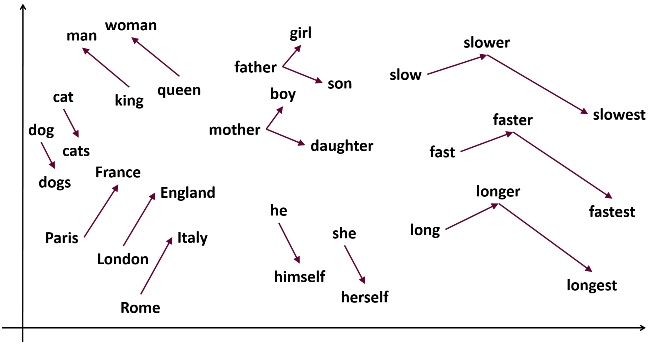
- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.
- Many ways to measure similarity: number of sentences both occur in, number of times both appear in the same window of w words, in similar positions of documents in different languages, etc.

LSA gives a way of embedding words into k-dimensional space.

- Embedding is via low-rank approximation of $\mathbf{X}^T\mathbf{X}$: where $(\mathbf{X}^T\mathbf{X})_{a,b}$ is the number of documents that both $word_a$ and $word_b$ appear in.
- Think about $\mathbf{X}^T\mathbf{X}$ as a similarity matrix (gram matrix, kernel matrix) with entry (a, b) being the similarity between $word_a$ and $word_b$.
- Many ways to measure similarity: number of sentences both occur in, number of times both appear in the same window of w words, in similar positions of documents in different languages, etc.
- Replacing $\mathbf{X}^T\mathbf{X}$ with these different metrics (sometimes appropriately transformed) leads to popular word embedding algorithms: word2vec, GloVe, fastText, etc.

EXAMPLE: WORD EMBEDDING





Note: word2vec is typically described as a neural-network method, but it is really just low-rank approximation of a specific similarity matrix. *Neural word embedding as implicit matrix factorization*, Levy and Goldberg.