Data representations and machine learning algorithms

Nikolaos Vasiloglou II
About a Nick
Audience statistics

- students/academics: 4
- software engineers: 9
- scientists: 5
- managers/consultants: 7
Outline

1. Some common data problems
2. Representation models
3. Storage models
4. Algorithms for sparse data
5. Algorithms for dense data
6. Book chaos in machine learning
An easy life for the data scientist

- Can you please run kmeans++ with k=4?
- Please run a Gaussian SVM with c=5 and bandwidth=10
- Can you tell me what are the 3 first eigenvalues of the covariance matrix?
- Can you run spectral partitioning on the graph?
The reality is different. Mapping the real problems to numbers

- Text data
- User preference
- Web clicks
- IP traffic
- Social network relationship data
- Market Basket data
- Sales volume data
- Descriptive data
- Images
- Video
The most generic data representation, the **bipartite graph**
Relational data = pairwise relationships = full graph

From a bipartitite graph to a full graph
Sales volume

\[
\text{Today\_Volume} \sim \text{previous\_volume} + \text{temperature} + \text{humidity} + \text{rack\_location} + \text{is\_holiday}
\]
John likes to watch movies. Mary likes too. 
John also likes to watch football games.

dictionary={1:"John", 2:"likes", 3:"to", 4:"watch", 5:"movies", 6:"also", 7:"football", 8:"games", 9:"Mary", 10:"too"},

[1, 2, 1, 1, 1, 0, 0, 1, 1, 1]  
[1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

D1 "I like databases"  
D2 "I hate databases"

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>like</th>
<th>hate</th>
<th>databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Bag of words on images and more.

You can always build dictionaries on any type of data, such as images.
### User ratings

#### Netflix data

- **480,000 users**
- **18,000 movies**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>x</th>
<th>...</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>5</td>
<td>...</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3</td>
<td>x</td>
<td>...</td>
</tr>
<tr>
<td>x</td>
<td>4</td>
<td>3</td>
<td>x</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>...</td>
</tr>
<tr>
<td>x</td>
<td>5</td>
<td>x</td>
<td>1</td>
<td>...</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>3</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>...</td>
</tr>
</tbody>
</table>
IP traffic

ip to ip data volume

\[ X = \begin{pmatrix} x_1 & x_2 & \cdots \\ x_3 & x_4 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \]
## Descriptive data

<table>
<thead>
<tr>
<th>customer\feature</th>
<th>age</th>
<th>income</th>
<th>total purchases</th>
<th>hair color</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td>19</td>
<td>40000</td>
<td>500</td>
<td>gray</td>
</tr>
<tr>
<td>nick</td>
<td>45</td>
<td>55000</td>
<td>450</td>
<td>black</td>
</tr>
<tr>
<td>peter</td>
<td>22</td>
<td>12000</td>
<td>1890</td>
<td>brown</td>
</tr>
<tr>
<td>joanna</td>
<td>25</td>
<td>90000</td>
<td>7000</td>
<td>brown</td>
</tr>
</tbody>
</table>
Storage models

- Key:value [0:22.3, 22:2.23, 44:3.12,...]
- Rectangular grid [0.2,0.33,3.22,..]

- Dense data => rectangular grid
- Sparse data => key:value or compressed row/column format
Dense data

Advantages of rectangular grid versus key-value

- Compact memory
- Better memory locality
- Optimized HPC libraries like BLAS/LAPACK
- Avoid unnecessary joins
- Dynamically adding rows is easy
Sparse Data

- ARPACK compressed row/column format
- Optimized Linear Algebra libraries
- Hard to dynamically add data
- Key-value store works well
- BLAS level operations between Sparse-Sparse Vectors/Matrices are slow
- Sparse-Dense are fast and are the most common ones
Algorithms for dense data

- Euclidean distance
  \[ d(x,y) = \|x-y\|^2 = \sum (x_i-y_i)^2 \]
- Gaussian Kernel \( \exp(d(x,y)/h) \)
- Dot Products \( \langle x, y \rangle = \sum x_i y_i \)
- Vector additions/subtractions \( x+y \)
- Mean normalization \( x = x - \mu_x \)
- Variance normalization \( x = x / \sigma_x \)
- Matrix Matrix Multiplication \( A = B \times C \)
Algorithms for sparse data

- Dot products $<x,y> = \Sigma x_i y_i$
- Euclidean distances NO
- Matrix Vector multiplication $A = Bx$
- Mean normalization NO
- Variance/Norm normalization Yes
- Matrix Matrix Multiplication NO
- Linear methods are a must
What does 0 mean in dense and in sparse data?

Dense data

<table>
<thead>
<tr>
<th>people\course</th>
<th>math</th>
<th>literature</th>
<th>history</th>
<th>physics</th>
</tr>
</thead>
<tbody>
<tr>
<td>nick</td>
<td>95</td>
<td>45</td>
<td>99</td>
<td>22</td>
</tr>
<tr>
<td>john</td>
<td>17</td>
<td>0</td>
<td>14</td>
<td>75</td>
</tr>
<tr>
<td>george</td>
<td>44</td>
<td>89</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>amy</td>
<td>33</td>
<td>89</td>
<td>57</td>
<td>86</td>
</tr>
<tr>
<td>cathy</td>
<td>66</td>
<td>34</td>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>jennifer</td>
<td>77</td>
<td>55</td>
<td>87</td>
<td>12</td>
</tr>
<tr>
<td>bob</td>
<td>12</td>
<td>99</td>
<td>65</td>
<td>0</td>
</tr>
</tbody>
</table>
What does 0 mean in dense and in sparse data?

<table>
<thead>
<tr>
<th>user/page</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
<th>p6</th>
<th>p7</th>
<th>...</th>
<th>p100</th>
<th>p101</th>
<th>p102</th>
<th>p103</th>
<th>p104</th>
</tr>
</thead>
<tbody>
<tr>
<td>usr1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>usr2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>user/movie</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
<th>m6</th>
<th>m7</th>
<th>...</th>
<th>m100</th>
<th>m101</th>
<th>m102</th>
<th>m103</th>
<th>m104</th>
</tr>
</thead>
<tbody>
<tr>
<td>usr1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>usr2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Sparse data as a graph
When does a zero in sparse data always mean zero

In categorical data

<table>
<thead>
<tr>
<th>car\features</th>
<th>color</th>
<th>brand</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>car1</td>
<td>red</td>
<td>audi</td>
<td>4.3</td>
</tr>
<tr>
<td>car2</td>
<td>blue</td>
<td>audi</td>
<td>3.4</td>
</tr>
<tr>
<td>car3</td>
<td>green</td>
<td>ford</td>
<td>2.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>car\features</th>
<th>red</th>
<th>blue</th>
<th>green</th>
<th>audi</th>
<th>ford</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>car1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4.3</td>
</tr>
<tr>
<td>car2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3.4</td>
</tr>
<tr>
<td>car3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2.8</td>
</tr>
</tbody>
</table>
Not exactly there, a small correction

<table>
<thead>
<tr>
<th>car\features</th>
<th>red</th>
<th>blue</th>
<th>audi</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>car1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4.3</td>
</tr>
<tr>
<td>car2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.4</td>
</tr>
<tr>
<td>car3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.8</td>
</tr>
</tbody>
</table>

A simple rule. If you have N categories, expand it in N-1 columns
Converting Sparse data to Dense

- Also called clustering
- Achieved with matrix factorizations

\[
\begin{pmatrix}
    a_{11} & \ldots & a_{1n} \\
    \vdots & \ddots & \vdots \\
    a_{m1} & \ldots & a_{mn}
\end{pmatrix}
\begin{pmatrix}
    a_{11} & \ldots & a_{1k} \\
    \vdots & \ddots & \vdots \\
    a_{m1} & \ldots & a_{mk}
\end{pmatrix}
\begin{pmatrix}
    a_{11} & \ldots & a_{1n} \\
    \vdots & \ddots & \vdots \\
    a_{k1} & \ldots & a_{kn}
\end{pmatrix}
\]

\[
\begin{pmatrix}
    a_{11} & 0 & a_{13} \\
    a_{21} & a_{22} & 0 \\
    0 & a_{32} & 0
\end{pmatrix}
\begin{pmatrix}
    b_{11} & b_{12} \\
    b_{21} & b_{22} \\
    b_{31} & b_{33}
\end{pmatrix}
\begin{pmatrix}
    c_{11} & c_{12} & c_{13} \\
    c_{21} & c_{22} & c_{23}
\end{pmatrix}
\]
Why do we want to convert sparse data to dense?

- Summarization clustering
- Denoising
- Computing neighborhoods
- Information extraction
- Most popular methods
  - Singular Value Decomposition
  - NonNegative Matrix Factorization
Converting dense data to sparse

- Old good quantization
- For every column find the min and the max value
- Divide the range in q bins
- Split the column in q new ones
Why do we want to convert dense data to sparse?

- Linearize nonlinear data
- Sometimes sparse data can better summarize/compress dense high dimensional data
Linearizing nonlinear data

\[ \hat{y} = 7.79 - 0.066x_1 \]

\[ \hat{y} = 14.86 - 0.167x_1 \]
## An example

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>point1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>point2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>point3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>point4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>point5</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>point6</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

For the first 3 points the model is \( x = y \)

For the next 3 points the model is \( x = 0.5y \)
Now we have a unified linear model \[ x = y_1 + 0.5y_2 \]
Few words about the books

- Not for sale
- Not for lending
- They are divided in three categories
  - For business people
  - For practitioners
  - For scientists
For business people

● SuperCrunchers
● Competing with Analytics
● The lady tasting tea
● The theory that would not die
● Beautiful data
● The quest for artificial intelligence
For scientists

- Pattern classification
- Biological sequence analysis
- Bayesian Reasoning and machine learning
- Pattern Recognition and machine learning
- Recommender systems
- The quest for artificial intelligence
- Core concepts in data analysis
- Probabilistic graphical models
- Foundations of statistical language processing
- The elements of statistical learning
- Graph algorithms in the language of linear algebra
- Networks an introduction
For practitioners

- Temporal data mining
- Mining graph data
- Programming collective intelligence
- Data Analysis with open source tools
- Data mining
- Machine learning for hackers
- Collective intelligence in action
- Web data mining
- Data mining techniques in CRM
- Recommender systems
- Mining massive datasets
- Beautiful data
A publisher critique

- Cambridge University press: High quality, very academic, reasonable prices
- Springer: Large variety, high quality, very expensive
- MIT press: Limited but ok collection, low prices, very academic