Matrix Calculus

- Gradient descent for machine learning is most often carried out over *vector* arguments
- In this case we just differentiate component by component and sometimes this allows us to collect terms in a compact notation
- See some familiar derivative identities compared to single-variable calculus

$$\frac{\partial \mathbf{x}^T A \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T) \tag{1}$$

Matrix Calculus: Differentiating a scalar by a vector

- For some scalar quantity $u(\mathbf{x})$ that depends on a vector $\mathbf{x}$ the derivative is another vector of the partials with respect to each component

$$\frac{\partial u}{\partial \mathbf{x}} = \left[\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}, \ldots, \frac{\partial u}{\partial x_n}\right] \tag{2}$$

- What is the derivative of a vector with respect to another vector?

Matrix Calculus: chain and product rules

- Same basic rules of differentiation apply when we differentiate with respect to matrices or vectors

$$\frac{\partial \mathbf{u} \cdot \mathbf{v}}{\partial \mathbf{x}} = ? \tag{3}$$

- Derivative of a dot product of two vectors u and v that may depend on a third x? What rule might we want to apply here?

Matrix notation for large problems

- A practical machine learning problem may involve vectors and matrices of large dimension
- Matrix notation allows us a convenient method of keeping track of all of the independent parameters and express the update in a convenient fashion
- For example to do least squares for a simple matrix problem we find $|\mathbf{Ax} - \mathbf{b}|^2$

$$\nabla \text{ w.r.t } \mathbf{x} = 2A^T(Ax - b) \tag{4}$$

Singular Value Decomposition

$$A = USV^T \tag{5}$$

- The singular value decomposition of a matrix expresses it as a product of three other matrices U, S, and V^T
- U and V are square orthogonal (unitary) matrices: $UU^T = I$
- S is a diagonal but not necessarily square matrix
- The diagonal entries of S, $s_{ii} = \sigma_i$ are called the singular values
- Note: The singular values are equal to the square roots of the eigenvalues of $A^T A$

Low-Rank Approximations (1)

- For us, the important property of the SVD is that it provides a convenient tool for creating low-rank approximations of our original matrix or data

$$A \approx \hat{A}_k \tag{6}$$

- The SVD is an optimal in terms of the *Frobenius Norm* of the error $A - \hat{A}_k$ over the set of all rank k approximations

Low-Rank Approximations (2)

- To actually create the approximation we zero out the $n - k$ smallest singular values in S

$$A = U \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots \\ 0 & \sigma_2 & 0 & 0 & \cdots \\ 0 & 0 & \sigma_3 & 0 & \cdots \\ 0 & 0 & 0 & \sigma_4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} V^T \tag{7}$$

$$\hat{A}_2 = U \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots \\ 0 & \sigma_2 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{0} \equiv \sigma_3 & 0 & \cdots \\ 0 & 0 & 0 & \mathbf{0} \equiv \sigma_4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} V^T \tag{8}$$

SVD and PCA

- The low-rank approximations provided by singular value decomposition are conceptually similar to the process of principal component analysis
- Goal of *Principal Component Analysis*: Find a projection into a lower dimensional space that preserves meaningful relationships between data-instances and allows reconstruction with low error

Document-Term Matrix

- Matrix consisting of word frequencies over an entire set (corpus) of documents
- Row corresponds to a document
- Column corresponds to a word

$$\text{document term matrix} = \begin{bmatrix} \cdot & \text{term1} & \text{term2} & \text{term3} & \cdots \\ \text{doc1} & 1 & 0 & 1 & \cdots \\ \text{doc2} & 1 & 0 & 0 & \cdots \\ \text{doc3} & 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{9}$$

Bag-of-words and tf-idf

Bag-of-words

Representation of a text document by a vector consisting of word frequencies or a quantity derived from it

$$X_i = [x_{1i}, x_{2i}, \ldots, x_{mi}] \tag{10}$$

$$x_{ji} = t_{ji} \cdot \log\left(\frac{n}{idf_j}\right) \tag{11}$$

- term frequency (t), total number of documents (n), number of documents that contain the term (idf_j)

Latent Semantic Analysis

- Latent Semantic Analysis (LSA) [Introduction to Latent Semantic Analysis, Landauer *et al* 1998] or Latent Semantic Indexing is basically the combination of the analysis of a document corpus in terms of word frequencies and the singular value decomposition
- More powerful mathematical analysis compared to word co-occurences or usage correlations
- Capable of inferring "deeper relations" between different terms and documents

Latent Semantic Analysis (2)

- The dimensionality reduction entailed by the SVD is the crucial (step) in reproducing a representation closer to human cognitive relations
- Approximation can be thought of as a kind of averaging process: term is an average of the meaning of the documents it appears in, document is an average of the terms it contains [Landauer]
- The SVD jointly derives both of these types of representations

Latent Semantic Analysis (3)

- Compare documents and terms in the space provided by the SVD approximation
- Any two vectors can be compared by looking at their cosine similarity, angle between them
- The main idea of LSA is that it is better to compare words and documents in the low rank space

Document Term Matrix for Paper Titles

Example of text data: Titles of Some Technical Memos

c1:	*Human* machine *interface* for ABC *computer* applications
c2:	A *survey* of *user* opinion of *computer system response time*
c3:	The *EPS user interface* management *system*
c4:	*System* and *human system* engineering testing of *EPS*
c5:	Relation of *user* perceived *response time* to error measurement
m1:	The generation of random, binary, ordered *trees*
m2:	The intersection *graph* of paths in *trees*
m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4:	*Graph minors*: A *survey*

$\{X\} =$

	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

- Document-Term Matrix from Landauer et al

LSA for Paper Titles

$$\{\hat{X}\} =$$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

$\underline{r}$(human.user) = .94

$\underline{r}$ (human.minors) = -.83

- Matrix Obtained after LSA

Semantic Priming

- “The townspeople were amazed to find that all the buildings had collapsed except the *mint*.”
- “Thinking of the amount of garlic in his dinner, the guest asked for a *mint*.”
- “The husband was afraid that his jealous wife would discover his new interest.”
- “The millionaire jumped from the window when he heard about the new rate of interest.”
- Priming study by Till, Mross, and Kintsch

Semantic Priming

- Expectations allow us to recognize things more easily
- "The townspeople were amazed to find that all the buildings had collapsed except the *mint*."
- "Thinking of the amount of garlic in his dinner, the guest asked for a *mint*."
- Target words: money, candy, earthquake, breath, ground

Text Mining Tools in Python

- Execute the steps of LSA on a toy document corpus
- Find the SVD approximation for each possible rank
- The tokenizers for counts and tf-idf are found in `sklearn.feature_extraction.text`
- Do simple LSA with `np.linalg.svd` and compare with your expectations

Code and List

```
corpus = ['To be, or not to be, that is the question',
           'Whether tis nobler in the mind to suffer',
           'The slings and arrows of outrageous fortune',
           'Or to take arms against a sea of troubles',
           'And by doing something',
           'the the the the the the the'
]

vectorizer = CountVectorizer(min_df=1)

dt = vectorizer.fit_transform(corpus)

x = vectorizer.get_feature_names();
```

- Use the vectorizer classes to transform a collection of documents (array of strings) into a document-term matrix

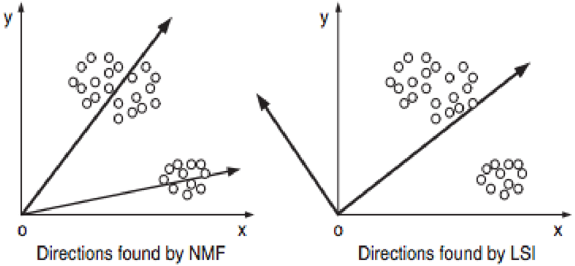
Non-negative Matrix Factorization

- Non-negative Matrix Factorization provides a similar decomposition of the dt matrix
- Naturally additive model since the values are all constrained to be non-negative

$$\arg\min_{W,H} ||X - WH||^2 = \sum_{i,j} X_{ij} - WH_{ij}$$

Document Clustering

- *Document Clustering*: Partitioning a corpus into a predefined number of clusters related to a coherent topic
- NMF applied to text analysis by Xu, Liu, Gong "Document Clustering Base on Non-negative Matrix Factorization"



NMF Document Clustering Algorithm

- *Document Clustering Algorithm*:
  - Construct the term-document matrix $\mathbf{X}$ from the given corpus
  - Find an NMF decomposition of $\mathbf{X}$
  - Normalize the factors U and V
  - Examine each column of V and look for the component with the largest value and assign the corresponding document to cluster k
- Standard Datasets for Document Clustering: NIST Topic Detection and Tracking (TDT2), Reuters dataset

Parts based decomposition

- Some of the original NMF papers by Lee and Seung showed that the non-negativity constraint favored a "parts based decomposition" of the dataset