Logistics

- Assignment 2’s q4 and q5 is due at 6pm on 03/02.

- Project progress report is due on 03/10, no hard copy is needed.
Project progress report

• What changes you have made for the task
  – No change at all
  – Change the data, objective, or else

• Describe data preprocessing
  – What are the features or representations?
    • Numerical or categorical?
  – Do you use all the data or part of it?

• What method you have tried
  – E.g logistic regression, SVM
Project progress report

• What results do you have now?
  – Baselines?
  – Evaluation metrics?
    • Precision, recall, F1, accuracy
  – How are your results compared to the baselines?
Baselines

• Roughly speaking, baselines are some methods that are easy to come up with
  – Classification
    • Random baseline: randomly generate the predictions
    • Majority baseline: the majority class in the training data
    • Easy features: raw pixels, single words
Example

• Text categorization
  – Training data: 80 regular emails, 20 spam emails
  – Labels: \{regular, spam\}
  – Random baseline: 80/100 regular, 20/100 spam
  – Majority baseline: \(p=1.0\) regular
  – Easy features: unigrams
Evaluation Measures

- Accuracy
- Precision/recall/f-measure
- ROC

[Some slides are borrowed from Rich Caruana]
Accuracy

• Target: 0/1, -1/+1, True/False, ...
• Accuracy := #correct/#total prediction
Confusion Matrix

\[
\begin{array}{cc}
\text{Predicted 1} & \text{Predicted 0} \\
\hline \\
\text{True 1} & a & b \\
\text{True 0} & c & d \\
\end{array}
\]

accuracy = (a+d) / (a+b+c+d)
## Prediction Threshold

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>True 0</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

- threshold > MAX(f(x))
- all cases predicted 0
- (b+d) = total
- accuracy = %False = %0’s

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>True 0</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

- threshold < MIN(f(x))
- all cases predicted 1
- (a+c) = total
- accuracy = %True = %1’s
82% 0’s in data

18% 1’s in data

optimal threshold
Problems with Accuracy

• Assumes equal cost for both kinds of errors
  – Medical domain: cold v.s. cancer

• is 99% accuracy good?
  – can be excellent, good, mediocre, poor, terrible

• is 10% accuracy bad?
  – information retrieval

• BaseRate = accuracy of predicting predominant class
Percent Reduction in Error

- 80% accuracy = 20% error
- suppose learning increases accuracy from 80% to 90%
- error reduced from 20% to 10%
- 50% reduction in error

- 99.90% to 99.99% = 90% reduction in error
- 50% to 75% = 50% reduction in error
- can be applied to many other measures
**Costs (Error Weights)**

<table>
<thead>
<tr>
<th></th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>$W_a$</td>
<td>$W_b$</td>
</tr>
<tr>
<td>True 0</td>
<td>$W_c$</td>
<td>$W_d$</td>
</tr>
</tbody>
</table>

- Often $W_a = W_d = \text{zero}$ and $W_b \neq W_c \neq \text{zero}$
Precision and Recall

- typically used in document retrieval
- Precision:
  - how many of the returned documents are correct
  - precision(threshold)
- Recall:
  - how many of the positives does the model return
  - recall(threshold)
- Precision/Recall Curve: sweep thresholds
Summary Stats: F & BreakEvenPt

\[
\text{PRECISION} = \frac{a}{a + c}
\]

\[
\text{RECALL} = \frac{a}{a + b}
\]

\[
F = \frac{2 \times (\text{PRECISION} \times \text{RECALL})}{(\text{PRECISION} + \text{RECALL})}
\]

BreakEvenPoint = PRECISION = RECALL

harmonic average of precision and recall
F Scores: 0.6210 (0.5); 0.6627 (freq); 0.6190 (max_acc)

Break Even Point = 0.6627

Better performance

Worse performance

Precision

Recall

y = x
**Confusion Matrix**

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td><strong>false positive</strong></td>
</tr>
<tr>
<td><strong>False 0</strong></td>
<td><strong>false positive</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>TP</td>
</tr>
<tr>
<td><strong>False 0</strong></td>
<td>FP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>$P(pr1</td>
</tr>
<tr>
<td><strong>False 0</strong></td>
<td>$P(pr1</td>
</tr>
</tbody>
</table>

**Definitions:**
- **True Positive (TP)**: Correctly predicted positive cases.
- **False Positive (FP)**: Incorrectly predicted positive cases.
- **True Negative (TN)**: Correctly predicted negative cases.
- **False Negative (FN)**: Incorrectly predicted negative cases.
- **Hits**: Correctly predicted positive cases.
- **Misses**: Incorrectly predicted positive cases.
- **False Alarms**: Incorrectly predicted negative cases.
- **Correct Rejections**: Correctly predicted negative cases.

**Probabilities:**
- $P(pr1|tr1)$: Probability of predicting 1 given true 1.
- $P(pr0|tr1)$: Probability of predicting 0 given true 1.
- $P(pr1|tr0)$: Probability of predicting 1 given true 0.
- $P(pr0|tr0)$: Probability of predicting 0 given true 0.
ROC Plot and ROC Area

- Receiver Operator Characteristic
- Developed in WWII to statistically model false positive and false negative detections of radar operators
- Better statistical foundations than most other measures
- Standard measure in medicine and biology
- Becoming more popular in ML
ROC Plot

- Sweep threshold and plot
- Sensitivity = \( \frac{a}{a+b} = \text{Recall} \)
  - True positive rate
- \( 1 - \text{Specificity} = 1 - \frac{d}{c+d} = \frac{c}{c+d} \)
  - False positive rate

<table>
<thead>
<tr>
<th></th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td><strong>True 0</strong></td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>
Sensitivity = True Positive Rate = $P(\text{pred_true}|\text{true})$

ROC Area = 0.9049

diagonal line is random prediction
Properties of ROC

- ROC Area:
  - 1.0: perfect prediction
  - 0.9: excellent prediction
  - 0.8: good prediction
  - 0.7: mediocre prediction
  - 0.6: poor prediction
  - 0.5: random prediction
  - <0.5: something wrong!
Properties of ROC

• Slope is non-increasing
• Each point on ROC represents different tradeoff (cost ratio) between false positives and false negatives
• Slope of line tangent to curve defines the cost ratio
• ROC Area represents performance averaged over all possible cost ratios
• If two ROC curves do not intersect, one method dominates the other
• If two ROC curves intersect, one method is better for some cost ratios, and other method is better for other cost ratios
Summary

- the measure you optimize to makes a difference
- the measure you report makes a difference
- use measure appropriate for problem/community
- accuracy often is not sufficient/appropriate
- ROC is gaining popularity in the ML community
- only accuracy generalizes to >2 classes!
What we learned last time

• Unsupervised learning for neural networks
  – Restricted Boltzmann machine
  – Autoencoders
  – Sparse coding
RESTRICTED BOLTZMANN MACHINE

Topics: RBM, visible layer, hidden layer, energy function

Energy function:
\[ E(x, h) = -h^T W x - c^T x - b^T h \]

\[ = - \sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j \]

Distribution:
\[ p(x, h) = \exp(-E(x, h))/Z \]

partition function (intractable)
Training RBM

**Topics:** training objective

- To train an RBM, we’d like to minimize the average negative log-likelihood (NLL)

\[ \frac{1}{T} \sum_t l(f(x^{(t)})) = \frac{1}{T} \sum_t -\log p(x^{(t)}) \]

- We’d like to proceed by stochastic gradient descent

\[ \frac{\partial}{\partial \theta} - \log p(x^{(t)}) \bigg|_{x^{(t)}} = E_h \left[ \frac{\partial E(x^{(t)}, h)}{\partial \theta} \bigg|_{x^{(t)}} \right] - E_{x,h} \left[ \frac{\partial E(x, h)}{\partial \theta} \right] \]

positive phase  
negative phase
**CONTRASTIVE DIVERGENCE (CD)**

(HINTON, NEURAL COMPUTATION, 2002)

**Topics:** contrastive divergence, negative sample

- **Idea:**
  1. replace the expectation by a point estimate at $\tilde{x}$
  2. obtain the point $\tilde{x}$ by Gibbs sampling
  3. start sampling chain at $x^{(t)}$
**CONTRASTIVE DIVERGENCE (CD)**

*(HINTON, NEURAL COMPUTATION, 2002)*

**Topics:** contrastive divergence, negative sample

\[
E_h \left[ \frac{\partial E(x^{(t)}, h)}{\partial \theta} \middle| x^{(t)} \right] \approx \frac{\partial E(x^{(t)}, \tilde{h}^{(t)})}{\partial \theta} \quad \text{and} \quad E_{x,h} \left[ \frac{\partial E(x, h)}{\partial \theta} \right] \approx \frac{\partial E(\tilde{x}, \tilde{h})}{\partial \theta}
\]

![Graph showing contrastive divergence](image)
CD-K: PSEUDOCODE

**Topics:** contrastive divergence

1. For each training example \( x^{(t)} \)
   i. generate a negative sample \( \tilde{x} \) using
      
      \[ k \text{ steps of Gibbs sampling, starting at } x^{(t)} \]
   ii. update parameters
      
      \[
      W \leftarrow W + \alpha \left( h(x^{(t)}) x^{(t)^T} - h(\tilde{x}) \tilde{x}^T \right)
      \]
      
      \[
      b \leftarrow b + \alpha \left( h(x^{(t)}) - h(\tilde{x}) \right)
      \]
      
      \[
      c \leftarrow c + \alpha \left( x^{(t)} - \tilde{x} \right)
      \]

2. Go back to 1 until stopping criteria
<table>
<thead>
<tr>
<th></th>
<th>Bias Unit</th>
<th>Hidden 1</th>
<th>Hidden 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias Unit</td>
<td>-0.08257658</td>
<td>-0.19041546</td>
<td>1.57007782</td>
</tr>
<tr>
<td>Harry Potter</td>
<td>-0.82602559</td>
<td>-7.08986885</td>
<td>4.96606654</td>
</tr>
<tr>
<td>Avatar</td>
<td>-1.84023877</td>
<td>-5.18354129</td>
<td>2.27197472</td>
</tr>
<tr>
<td>LOTR 3</td>
<td>3.92321075</td>
<td>2.51720193</td>
<td>4.11061383</td>
</tr>
<tr>
<td>Gladiator</td>
<td>0.10316995</td>
<td>6.74833901</td>
<td>-4.00505343</td>
</tr>
<tr>
<td>Titanic</td>
<td>-0.97646029</td>
<td>3.25474524</td>
<td>-5.59606865</td>
</tr>
<tr>
<td>Glitter</td>
<td>-4.44685751</td>
<td>-2.81563804</td>
<td>-2.91540988</td>
</tr>
</tbody>
</table>
Topics: autoencoder, encoder, decoder, tied weights

- Feed-forward neural network trained to reproduce its input at the output layer.

\[
\hat{x} = o(\hat{a}(x)) = \text{sigmoid}(c + W^* h(x))
\]

for binary inputs

\[
h(x) = g(a(x)) = \text{sigmoid}(b + Wx)
\]

Just like a feed-forward NN!
**Topics:** undercomplete representation

- **Hidden layer is undercomplete if smaller than the input layer**
  - hidden layer “compresses” the input
  - will compress well only for the training distribution
- **Hidden units will be**
  - good features for the training distribution
  - but bad for other types of input

\[
\begin{align*}
\hat{x} & \quad c_k \\
\text{W}^* &= \text{W}^T \\
h(x) & \quad b_j \\
x & \quad \text{W}
\end{align*}
\]
DENOISING AUTOENCODER

**Topics:** denoising autoencoder

- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability $\nu$
  - Gaussian additive noise
- Reconstruction $\hat{x}$ computed from the corrupted input $\tilde{x}$
- Loss function compares $\hat{x}$ reconstruction with the **noiseless input** $x$
**Topics:** denoising autoencoder

- Idea: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability \( \nu \)
  - Gaussian additive noise
- Reconstruction \( \mathbf{\hat{x}} \) computed from the corrupted input \( \mathbf{\tilde{x}} \)
- Loss function compares \( \mathbf{\hat{x}} \) reconstruction with the noiseless input \( \mathbf{x} \)

\[
\mathbf{W}^* = \mathbf{W}^T \\
(\text{tied weights})
\]

\[
\mathbf{0}
\]

\[
p(\mathbf{\tilde{x}}|\mathbf{x})
\]
FILTERS (DENOISING AUTOENCODER)
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

• No corrupted inputs (cross-entropy loss)
FILTERS (DENOISING AUTOENCODER)
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

- 25% corrupted inputs
FILTERS (DENOISING AUTOENCODER)
(Vincent, Larochelle, Bengio and Manzagol, ICML 2008)

• 50% corrupted inputs
With more noisy input, the model needs more information (e.g. more pixels) to identify the “features”
**Sparse Coding**

**Topics:** sparse coding

- For each $x^{(t)}$ find a latent representation $h^{(t)}$ such that:
  - it is sparse: the vector $h^{(t)}$ has many zeros
  - we can reconstruct the original input $x^{(t)}$ as well as possible

- More formally:

$$
\min_D \frac{1}{T} \sum_{t=1}^{T} \min_{h^{(t)}} \frac{1}{2} \| x^{(t)} - D h^{(t)} \|_2^2 + \lambda \| h^{(t)} \|_1
$$

- $D$ is equivalent to the autoencoder output weight matrix
- however, $h(x^{(t)})$ is now a complicated function of $x^{(t)}$
  - encoder is the minimization $h(x^{(t)}) = \arg\min_{h^{(t)}} \frac{1}{2} \| x^{(t)} - D h^{(t)} \|_2^2 + \lambda \| h^{(t)} \|_1$
Why sparse coding?

Topics: dictionary

- Can also write \( \hat{x}^{(t)} = D \cdot h(x^{(t)}) = \sum_{k \text{ s.t. } h(x^{(t)})_k \neq 0} D_{.,k} h(x^{(t)})_k \)

we also refer to \( D \) as the dictionary

- in certain applications, we know what dictionary matrix to use
- often however, we have to learn it
**Topics:** dictionary

- Can also write \( \hat{x}^{(t)} = D \ h(x^{(t)}) = \sum_{k \text{ s.t. } h(x^{(t)})_k \neq 0} D_{:,k} h(x^{(t)})_k \)

\[
7 = 1 \begin{array}{c}
\text{0}
\end{array} + 1 \begin{array}{c}
\text{4}
\end{array} + 1 \begin{array}{c}
\text{7}
\end{array} + 1 \begin{array}{c}
\text{4}
\end{array} + 1 \begin{array}{c}
\text{7}
\end{array} + 1 \begin{array}{c}
\text{7}
\end{array} + 1 \begin{array}{c}
\text{7}
\end{array} + 1 \begin{array}{c}
\text{7}
\end{array} + 0.8 \begin{array}{c}
\text{4}
\end{array} + 0.8 \begin{array}{c}
\text{7}
\end{array} \]

- we also refer to \( D \) as the dictionary
  - in certain applications, we know what dictionary matrix to use
  - often however, we have to learn it
SPARSE CODING

**Topics:** dictionary

- Can also write \( \hat{x}^{(t)} = D \ h(x^{(t)}) = \sum_{k \text{ s.t.} \ h(x^{(t)})_k \neq 0} D_{.,k} \ h(x^{(t)})_k \)

\[
\begin{align*}
7 &= 1 \ + 1 \ + 1 \ + 1 \ + 1 \\
&\quad + 1 \ + 1 \ + 0.8 \ + 0.8 \ + 1 \\
\end{align*}
\]

- we also refer to \( D \) as the dictionary
  - in certain applications, we know what dictionary matrix to use
  - often however, we have to learn it
The full learning algorithm

Topics: learning algorithm (putting it all together)
• Learning alternates between inference and dictionary learning

• While D has not converged
  ‣ find the sparse codes $h(x^{(t)})$ for all $x^{(t)}$ in my training set with ISTA
  ‣ update the dictionary:
    - $A \leftarrow \sum_{t=1}^{T} x^{(t)} h(x^{(t)})^T$
    - $B \leftarrow \sum_{t=1}^{T} h(x^{(t)}) h(x^{(t)})^T$
    - run block-coordinate descent algorithm to update $D$
Today’s Outline

• Baselines and evaluation metrics

• Dimension reduction (or feature reduction)
  – Principal component analysis (PCA)
  – Singular value decomposition (SVD)
What is feature reduction?

- Feature reduction refers to the mapping of the original high-dimensional data onto a lower-dimensional space.
  - Criterion for feature reduction can be different based on different problem settings.
    - Unsupervised setting: minimize the information loss
    - Supervised setting: maximize the class discrimination

- Given a set of data points of p variables \( \{x_1, x_2, \cdots, x_n\} \)
  Compute the linear transformation (projection)

\[
G \in \mathbb{R}^{p \times d} : x \in \mathbb{R}^p \rightarrow y = G^T x \in \mathbb{R}^d \quad (d << p)
\]

[Some slides are borrowed from Jieping Ye]
What is feature reduction?

Original data

Reduced data

\[ G^T \in \mathbb{R}^{d \times p} \]

\[ X \in \mathbb{R}^p \]

\[ G \in \mathbb{R}^{p \times d} : X \rightarrow Y = G^T X \in \mathbb{R}^d \]
High-dimensional data

Gene expression

Face images

Handwritten digits
Why feature reduction?

• Most machine learning and data mining techniques may not be effective for high-dimensional data
  – Curse of Dimensionality
  – Query accuracy and efficiency degrade rapidly as the dimension increases.

• The intrinsic dimension may be small.
  – For example, the number of genes responsible for a certain type of disease may be small.
Why feature reduction?

- **Visualization**: projection of high-dimensional data onto 2D or 3D.

- **Data compression**: efficient storage and retrieval.

- **Noise removal**: positive effect on query accuracy.
Application of feature reduction

- Face recognition
- Handwritten digit recognition
- Text mining
- Image retrieval
Feature reduction algorithms

• Unsupervised
  – Singular value decomposition (SVD)
  – Latent Semantic Indexing (LSI): truncated SVD
  – Independent Component Analysis (ICA)
  – Principal Component Analysis (PCA)
  – Canonical Correlation Analysis (CCA)

• Supervised
  – Linear Discriminant Analysis (LDA)

• Semi-supervised
  – Research topic
What is Principal Component Analysis?

• Principal component analysis (PCA)
  – Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables
  – Retains most of the sample's information.
  – Useful for the compression and classification of data.

• By information we mean the variation present in the sample, given by the correlations between the original variables.
  – The new variables, called principal components (PCs), are **uncorrelated**, and are ordered by the fraction of the total information each retains.
Geometric picture of principal components (PCs)

- the 1\textsuperscript{st} PC $Z_1$ is a minimum distance fit to a line in $X$ space
- the 2\textsuperscript{nd} PC $Z_2$ is a minimum distance fit to a line in the plane perpendicular to the 1\textsuperscript{st} PC

PCs are a series of linear least squares fits to a sample, each orthogonal to all the previous.
Algebraic definition of PCs

Given a sample of \( n \) observations on a vector of \( p \) variables

\[
\{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^p
\]

define the first principal component of the sample by the linear transformation

\[
z_1 = a_1^T x_j = \sum_{i=1}^{p} a_{i1} x_{ij}, \quad j = 1, 2, \ldots, n.
\]

where the vector

\[
a_1 = (a_{11}, a_{21}, \ldots, a_{p1})
\]

\[
x_j = (x_{1j}, x_{2j}, \ldots, x_{pj})
\]

is chosen such that \( \text{var}[z_1] \) is maximum.
Algebraic derivation of PCs

To find $\mathbf{a}_1$ first note that

$$\text{var}[z_1] = E((z_1 - \bar{z}_1)^2) = \frac{1}{n} \sum_{i=1}^{n} (a_1^T x_i - a_1^T \bar{x})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} a_1^T (x_i - \bar{x})(x_i - \bar{x})^T a_1 = a_1^T S a_1$$

where $S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$ is the covariance matrix.

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the mean.

In the following, we assume the Data is centered. $\bar{x} = 0$
Algebraic derivation of PCs

Assume \( \bar{x} = 0 \)

Form the matrix: \( X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{p \times n} \)

then \( S = \frac{1}{n} XX^T \)
Algebraic derivation of PCs

To find $a_1$ that maximizes $\text{var}[z_1]$ subject to $a_1^T a_1 = 1$

Let $\lambda$ be a Lagrange multiplier

$$L = a_1^T S a_1 - \lambda (a_1^T a_1 - 1)$$

$$\frac{\partial}{\partial a_1} L = S a_1 - \lambda a_1 = 0$$

$$\Rightarrow (S - \lambda I_p) a_1 = 0$$

therefore $a_1$ is an eigenvector of $S$

corresponding to the largest eigenvalue $\lambda = \lambda_1$. 
To find the next coefficient vector \( a_2 \) maximizing \( \text{var}[z_2] \)

subject to \( \text{cov}[z_2, z_1] = 0 \)

and to \( a_2^T a_2 = 1 \)

First note that \( \text{cov}[z_2, z_1] = a_1^T Sa_2 = \lambda_1 a_1^T a_2 \)

then let \( \lambda \) and \( \phi \) be Lagrange multipliers, and maximize

\[
L = a_2^T Sa_2 - \lambda (a_2^T a_2 - 1) - \phi a_2^T a_1
\]
Algebraic derivation of PCs

\[ L = a_2^T S a_2 - \lambda (a_2^T a_2 - 1) - \phi a_2^T a_1 \]

\[ \frac{\partial}{\partial a_2} L = S a_2 - \lambda a_2 - \phi a_1 = 0 \Rightarrow \phi = 0 \]

\[ S a_2 = \lambda a_2 \quad \text{and} \quad \lambda = a_2^T S a_2 \]

Multiplied by \( a_1^T \) on the right.
Algebraic derivation of PCs

We find that \( a_2 \) is also an eigenvector of \( S \) whose eigenvalue \( \lambda = \lambda_2 \) is the second largest.

In general

\[
\text{var}[z_k] = a_k^T S a_k = \lambda_k
\]

- The \( k^{\text{th}} \) largest eigenvalue of \( S \) is the variance of the \( k^{\text{th}} \) PC.

- The \( k^{\text{th}} \) PC \( z_k \) retains the \( k^{\text{th}} \) greatest fraction of the variation in the sample.
Algebraic derivation of PCs

• Main steps for computing PCs
  – Form the covariance matrix $S$.
  – Compute its eigenvectors: $\{a_i\}_{i=1}^p$
  – Use the first $d$ eigenvectors $\{a_i\}_{i=1}^d$ to form the $d$ PCs.
  – The transformation $G$ is given by
    $$G \leftarrow [a_1, a_2, \cdots, a_d]$$

A test point $x \in \mathbb{R}^p \rightarrow G^T x \in \mathbb{R}^d$. 
Optimality property of PCA

Dimension reduction

Original data

\[ X \in \mathbb{R}^{p \times n} \rightarrow G^T X \in \mathbb{R}^{d \times n} \]

\[ G^T X \in \mathbb{R}^{d \times n} \rightarrow \overline{X} = G(G^T X) \in \mathbb{R}^{p \times n} \]

Reconstruction

\[ Y = G^T X \in \mathbb{R}^{d \times n} \]

\[ G \in \mathbb{R}^{p \times d} \]

\[ \overline{X} \in \mathbb{R}^{p \times n} \]
Optimality property of PCA

Main theoretical result:

The matrix $G$ consisting of the first $d$ eigenvectors of the covariance matrix $S$ solves the following min problem:

$$
\min_{G \in \mathbb{R}^{p \times d}} \left\| X - G(G^T X) \right\|_F^2 \quad \text{subject to} \quad G^T G = I_d
$$

$$
\left\| X - \overline{X} \right\|_F^2
$$

reconstruction error

PCA projection minimizes the reconstruction error among all linear projections of size $d$. 
PCA for image compression

Original Image

d=1

d=2

d=4

d=8

d=16

d=32

d=64

d=100
function [V newX D] = myPCA(X)
    X = bsxfun(@minus, X, mean(X,1));       %# zero-center
    C = (X'*X)./(size(X,1)-1);             %# cov(X)

    [V D] = eig(C);
    [D order] = sort(diag(D), 'descend');   %# sort cols high to low
    V = V(:,order);

    newX = X*V(:,1:end);
end
Today’s Outline

• Baselines and evaluation metrics

• Dimension reduction (or feature reduction)
  – Principal component analysis (PCA)
  – Singular value decomposition (SVD)
Singular value decomposition (SVD)
**Dimensionality Reduction**

- **Assumption:** Data lies on or near a low $d$-dimensional subspace
- **Axes of this subspace are effective representation of the data**
<table>
<thead>
<tr>
<th>customer</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
<th>Su</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Inc.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DEF Ltd.</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GHI Inc.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>KLM Co.</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Smith</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Johnson</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Thompson</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Rank of a Matrix**

- **Q:** What is *rank* of a matrix $A$?
- **A:** Number of *linearly independent* columns of $A$

For example:
- Matrix $A = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank $r=2$

- **Why?** The first two rows are linearly independent, so the rank is at least 2, but all three rows are linearly dependent (the first is equal to the sum of the second and third) so the rank must be less than 3.

**Why do we care about low rank?**
- We can write $A$ as two “basis” vectors: $[1 \ 2 \ 1 \ [-2 \ -3 \ 1]$
- And new coordinates of : $[1 \ 0] \ [0 \ 1] \ [1 \ 1]$
Rank is “Dimensionality”

- **Cloud of points 3D space:**
  - Think of point positions as a matrix:
    \[
    \begin{bmatrix}
    1 & 2 & 1 \\
    -2 & -3 & 1 \\
    3 & 5 & 0
    \end{bmatrix}
    \]
  - 1 row per point:

- **We can rewrite coordinates more efficiently!**
  - Old basis vectors: \([1 0 0] [0 1 0] [0 0 1]\)
  - New basis vectors: \([1 2 1] [-2 -3 1]\)
  - Then A has new coordinates: [1 0]. B: [0 1], C: [1 1]
  - Notice: We reduced the number of coordinates!
Goal of dimensionality reduction is to discover the axis of data!

Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

By doing this we incur a bit of error as the points do not exactly lie on the line.
Why reduce dimensions?

- Discover hidden correlations/topics
  - Words that occur commonly together
- Remove redundant and noisy features
  - Not all words are useful
- Interpretation and visualization
- Easier storage and processing of the data
SVD - Definition

\[ A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T \]

- **A**: Input data matrix
  - \( m \times n \) matrix (e.g., \( m \) documents, \( n \) terms)
- **U**: Left singular vectors
  - \( m \times r \) matrix (\( m \) documents, \( r \) concepts)
- **\( \Sigma \)**: Singular values
  - \( r \times r \) diagonal matrix (strength of each ‘concept’)
    (\( r \) : rank of the matrix \( A \))
- **V**: Right singular vectors
  - \( n \times r \) matrix (\( n \) terms, \( r \) concepts)
\[ A \approx U \Sigma V^T = \sum_i \sigma_i u_i \circ v_i^T \]
$A \approx U \Sigma V^T = \sum_i \sigma_i u_i \circ v_i^T$
It is **always** possible to decompose a real matrix $A$ into $A = U \Sigma V^T$, where

- $U$, $\Sigma$, $V$: unique
- $U$, $V$: column orthonormal
  - $U^T U = I$; $V^T V = I$ ($I$: identity matrix)
  - (Columns are orthogonal unit vectors)
- $\Sigma$: diagonal
  - Entries (**singular values**) are positive, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq ... \geq 0$)

SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example: Users to Movies

\( \begin{bmatrix} 
\text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix} = \begin{bmatrix} 
\Sigma \\
V^T \\
U \\
\end{bmatrix} \)

"Concepts"
AKA Latent dimensions
AKA Latent factors

\( D = 3 \)
\( d = 2 \)
### SVD – Example: Users-to-Movies

- **A = U Σ Vᵀ** - example: Users to Movies

<table>
<thead>
<tr>
<th></th>
<th>SciFi</th>
<th>Romance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alien</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Serenity</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Casablanca</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Amelie</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\]

\[
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\]

\[
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
\]
**SVD – Example: Users-to-Movies**

\[ A = U \Sigma V^T \] - example: Users to Movies

<table>
<thead>
<tr>
<th></th>
<th>Matrix</th>
<th>Alien</th>
<th>Serenity</th>
<th>Casablanca</th>
<th>Amelie</th>
</tr>
</thead>
<tbody>
<tr>
<td>SciFi</td>
<td>1 1 1 0 0 0</td>
<td>0.13 0.02 -0.01</td>
<td>0.41 0.07 -0.03</td>
<td>0.55 0.09 -0.04</td>
<td>0.68 0.11 -0.05</td>
</tr>
<tr>
<td>Romance</td>
<td>3 3 3 0 0</td>
<td>0.15 -0.59 0.65</td>
<td>0.07 -0.73 -0.67</td>
<td>0.07 -0.29 0.32</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SciFi-concept</th>
<th>Romance-concept</th>
<th>SciFi</th>
<th>Romance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SciFi</td>
<td>12.4</td>
<td>12.4</td>
<td></td>
</tr>
<tr>
<td>Romance</td>
<td>9.5</td>
<td>9.5</td>
<td></td>
</tr>
</tbody>
</table>

The SVD of the matrix \( A \) can be approximated by multiplying the matrices \( U \), \( \Sigma \), and \( V \).
A = U \Sigma V^T - example:  

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
\]

\textit{U} is “user-to-concept” similarity matrix.
### SVD – Example: Users-to-Movies

**A = U \Sigma V^T** - example:

<table>
<thead>
<tr>
<th>SciFi-concept</th>
<th>SciFi</th>
<th>Alien</th>
<th>Serenity</th>
<th>Casablanca</th>
<th>Amelie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.13</td>
<td>0.02</td>
<td>-0.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.41</td>
<td>0.07</td>
<td>-0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.55</td>
<td>0.09</td>
<td>-0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.68</td>
<td>0.11</td>
<td>-0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>-0.59</td>
<td>0.65</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>-0.73</td>
<td>-0.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>-0.29</td>
<td>0.32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SciFi**
- **Romance**

```
| 1  1  1  0  0 |
| 3  3  3  0  0 |
| 4  4  4  0  0 |
| 5  5  5  0  0 |
| 0  2  0  4  4 |
| 0  0  0  5  5 |
| 0  1  0  2  2 |
```

```
| 0.56 | 0.59 | 0.56 | 0.09 | 0.09 |
| 0.12 | -0.02| 0.12 | -0.69| -0.69|
| 0.40 | -0.80| 0.40 | 0.09 | 0.09 |
```

“strength” of the SciFi-concept: 12.4
A = U Σ V^T - example:

V is “movie-to-concept” similarity matrix
SVD - Interpretation #1

'movies', 'users' and 'concepts':
- **U**: user-to-concept similarity matrix
- **V**: movie-to-concept similarity matrix
- **Σ**: its diagonal elements: 'strength' of each concept
Singular value decomposition (SVD)
SVD – Dimensionality Reduction

- Instead of using two coordinates \((x, y)\) to describe point locations, let’s use only one coordinate \((z)\)
- Point’s position is its location along vector \(v_1\)
- How to choose \(v_1\)? Minimize reconstruction error
**Goal:** Minimize the sum of reconstruction errors:

\[ \sum_{i=1}^{N} \sum_{j=1}^{D} \| x_{ij} - z_{ij} \|^2 \]

- where \( x_{ij} \) are the “old” and \( z_{ij} \) are the “new” coordinates

**SVD** gives ‘best’ axis to project on:
- ‘best’ = minimizing the reconstruction errors
- In other words, **minimum reconstruction error**
SVD - Interpretation #2

A = U \Sigma V^T - example:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
= \begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
\]
SVD - Interpretation #2

\[ A = U \Sigma V^T \] - example:
- \( U \Sigma \): Gives the coordinates of the points in the projection axis

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\]

Projection of users on the “Sci-Fi” axis
\((U \Sigma)^T\):

\[
\begin{bmatrix}
1.61 & 0.19 & -0.01 \\
5.08 & 0.66 & -0.03 \\
6.82 & 0.85 & -0.05 \\
8.43 & 1.04 & -0.06 \\
1.86 & -5.60 & 0.84 \\
0.86 & -6.93 & -0.87 \\
0.86 & -2.75 & 0.41
\end{bmatrix}
\]
More details

- **Q:** How exactly is dim. reduction done?

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32 \\
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix}
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09 \\
\end{bmatrix}
\]
More details

Q: How exactly is dim. reduction done?
A: Set smallest singular values to zero

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
= \begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\times \begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\times
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
\]
### SVD - Interpretation #2

**More details**

- **Q:** How exactly is dim. reduction done?
- **A:** Set smallest singular values to zero

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix} \approx \begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32 \\
\end{bmatrix} \times \begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix} \times \begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09 \\
\end{bmatrix}
\]
More details

- **Q:** How exactly is dim. reduction done?
- **A:** Set smallest singular values to zero
More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero
More details

Q: How exactly is dim. reduction done?
A: Set smallest singular values to zero

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\approx
\begin{bmatrix}
0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\
2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\
3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\
4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\
0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\
-0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\
0.32 & 0.23 & 0.32 & 2.01 & 2.01
\end{bmatrix}
\]

Frobenius norm:
\[
\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}
\]

\[
\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}
\]
is “small”
SVD – Best Low Rank Approx.

\[ A = U \Sigma V^T \]

B is best approximation of A

\[ B = U \Sigma V^T \]
**Theorem:**
Let \( A = U \Sigma V^T \) and \( B = U S V^T \) where
\( S = \) diagonal \( r \times r \) matrix with \( s_i = \sigma_i \) \((i=1...k)\) else \( s_i = 0 \)
then \( B \) is a best rank(\( B \))=\( k \) approx. to \( A \)

What do we mean by “best”:
- \( B \) is a solution to \( \min_B \| A-B \|_F \) where rank(\( B \))=\( k \)

\[
\begin{pmatrix}
  x_{11} & x_{12} & \cdots & x_{1n} \\
  x_{21} & x_{22} & \cdots & x_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{m1} & x_{m2} & \cdots & x_{mn}
\end{pmatrix}
= \begin{pmatrix}
  u_{11} & \cdots & u_{1n} \\
  \vdots & \ddots & \vdots \\
  u_{m1} & \cdots & u_{mn}
\end{pmatrix}
\begin{pmatrix}
  \sigma_{11} & 0 & \cdots & 0 \\
  0 & \sigma_{22} & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \sigma_{r,r}
\end{pmatrix}
\begin{pmatrix}
  v_{11} & \cdots & v_{1n} \\
  \vdots & \ddots & \vdots \\
  v_{m1} & \cdots & v_{mn}
\end{pmatrix}
\]

\[
\| A - B \|_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2}
\]
SVD - Interpretation #2

Equivalent:
‘spectral decomposition’ of the matrix:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
= \begin{bmatrix}
\vdots & \vdots \\
v_1 & u_2
\end{bmatrix}
\times
\begin{bmatrix}
\sigma_1 \\
\sigma_2
\end{bmatrix}
\times
\begin{bmatrix}
v_1 \\
v_2
\end{bmatrix}
\]
Equivalent:
‘spectral decomposition’ of the matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
\]

\[= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots \]

\[\text{n \times 1 \quad 1 \times m} \]

Assume: \( \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq 0 \)

Why is setting small \( \sigma_i \) to 0 the right thing to do?
Vectors \( \mathbf{u}_i \) and \( \mathbf{v}_i \) are unit length, so \( \sigma_i \) scales them.
So, zeroing small \( \sigma_i \) introduces less error.
Q: How many $\sigma$s to keep?
A: Rule-of-a-thumb:
keep 80-90% of ‘energy’ $= \sum_i \sigma_i^2$

Assume: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$
SVD - Complexity

- To compute SVD:
  - $O(nm^2)$ or $O(n^2m)$ (whichever is less)
- But:
  - Less work, if we just want singular values
  - or if we want first $k$ singular vectors
  - or if the matrix is sparse

- Implemented in linear algebra packages like
  - LINPACK, Matlab, SPlus, Mathematica ...
SVD - Conclusions so far

- **SVD**: $A = U \Sigma V^T$: unique
  - $U$: user-to-concept similarities
  - $V$: movie-to-concept similarities
  - $\Sigma$: strength of each concept

- **Dimensionality reduction**:  
  - keep the few largest singular values (80-90% of ‘energy’)  
  - SVD: picks up linear correlations
Relation to Eigen-decomposition

- SVD gives us:
  - $A = U \Sigma V^T$

- Eigen-decomposition:
  - $A = X \Lambda X^T$
    - $A$ is symmetric
    - $U$, $V$, $X$ are orthonormal ($U^T U = I$),
    - $\Lambda$, $\Sigma$ are diagonal

- Now let’s calculate:
  - $AA^T = U \Sigma V^T(U \Sigma V^T)^T = U \Sigma V^T(V \Sigma^T U^T) = U \Sigma \Sigma^T U^T$
  - $A^T A = V \Sigma^T U^T (U \Sigma V^T) = V \Sigma \Sigma^T V^T$

Shows how to compute SVD using eigenvalue decomposition!
SVD: Properties

- \( A A^T = U \Sigma^2 U^T \)
- \( A^T A = V \Sigma^2 V^T \)
- \( (A^T A)^k = V \Sigma^{2k} V^T \)
  - E.g.: \( (A^T A)^2 = V \Sigma^2 V^T V \Sigma^2 V^T = V \Sigma^4 V^T \)
- \( (A^T A)^k \sim v_1 \sigma_1^{2k} v_1^T \) for \( k \gg 1 \)
Case study: How to query?

- **Q**: Find users that like ‘Matrix’
- **A**: Map query into a ‘concept space’ – how?

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
= 
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32 \\
\end{bmatrix}
\times 
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix}
\times 
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09 \\
\end{bmatrix}
\]
Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$q = \begin{bmatrix}
5 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$

Project into concept space:
Inner product with each ‘concept’ vector $v_i$
Case study: How to query?

- **Q**: Find users that like 'Matrix'
- **A**: Map query into a 'concept space' – how?

**Project into concept space:**

Inner product with each 'concept' vector $v_i$
Case study: How to query?

Compactly, we have:

\[ q_{\text{concept}} = q \, V \]

E.g.:

\[
q = \begin{bmatrix}
5 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} \times \begin{bmatrix}
0.56 & 0.12 \\
0.59 & -0.02 \\
0.56 & 0.12 \\
0.09 & -0.69 \\
0.09 & -0.69
\end{bmatrix}
= \begin{bmatrix}
2.8 \\
0.6
\end{bmatrix}
\]

movie-to-concept similarities (V)

SciFi-concept
Case study: How to query?

- How would the user $d$ that rated (‘Alien’, ‘Serenity’) be handled?
  \[ d_{\text{concept}} = d \, V \]

E.g.:

\[ q = \begin{bmatrix} 0 & 4 & 5 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} \]

\[ \text{SciFi-concept} = \begin{bmatrix} 5.2 & 0.4 \end{bmatrix} \]
**Case study: How to query?**

- **Observation:** User $d$ that rated (‘Alien’, ‘Serenity’) will be similar to user $q$ that rated (‘Matrix’), although $d$ and $q$ have zero ratings in common!

\[
d = \begin{bmatrix}
0 & 4 & 5 & 0 & 0 \\
\end{bmatrix} \quad \rightarrow \quad \begin{bmatrix}
5.2 & 0.4 \\
\end{bmatrix}
\]

\[
q = \begin{bmatrix}
5 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \quad \rightarrow \quad \begin{bmatrix}
2.8 & 0.6 \\
\end{bmatrix}
\]

Zero ratings in common  \quad \text{Similarity} \neq 0
SVD: Drawbacks

- Optimal low-rank approximation in terms of Frobenius norm
- Interpretability problem:
  - A singular vector specifies a linear combination of all input columns or rows
- Lack of sparsity:
  - Singular vectors are dense!
SVD: Drawbacks

- Optimal low-rank approximation in terms of Frobenius norm
- Interpretability problem:
  - A singular vector specifies a linear combination of all input columns or rows
- Lack of sparsity:
  - Singular vectors are dense!
CUR decomposition

• In large-data applications, it is normal for the matrix A being decomposed to be very sparse
  – Documents
• With SVD, even if A is sparse, U and V will be dense
CUR Decomposition

- **Goal:** Express $A$ as a product of matrices $C,U,R$
  Make $\|A-C\cdot U\cdot R\|_F$ small
- **“Constraints”** on $C$ and $R$:

$$
\begin{pmatrix}
A \\
A
\end{pmatrix}
\approx
\begin{pmatrix}
\text{something}
\end{pmatrix}
\cdot
\begin{pmatrix}
U \\
R
\end{pmatrix}
$$

Frobenius norm:
$$
\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}
$$
CUR Decomposition

- **Goal:** Express $A$ as a product of matrices $C, U, R$
  - Make $\|A - C \cdot U \cdot R\|_F$ small
- **“Constraints”** on $C$ and $R$:

\[
\begin{pmatrix}
A
\end{pmatrix} \simeq \begin{pmatrix}
C
\end{pmatrix} \cdot \begin{pmatrix}
U
\end{pmatrix} \cdot \begin{pmatrix}
R
\end{pmatrix}
\]

Pseudo-inverse of the intersection of $C$ and $R$

**Frobenius norm:**

\[
\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}
\]
CUR: Provably good approx. to SVD

- Let:
  $A_k$ be the “best” rank $k$ approximation to $A$ (that is, $A_k$ is SVD of $A$)

**Theorem** [Drineas et al.]
CUR in $O(m \cdot n)$ time achieves

- $\|A - \text{CUR}\|_F \leq \|A - A_k\|_F + \varepsilon \|A\|_F$

with probability at least $1 - \delta$, by picking

- $O(k \log(1/\delta)/\varepsilon^2)$ columns, and
- $O(k^2 \log^3(1/\delta)/\varepsilon^6)$ rows

**In practice:**
Pick $4k$ cols/rows
CUR: How it Works

- Sampling columns (similarly for rows):

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size $c$

Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$  [column distribution]
2. $P(x) = \frac{\sum_i \mathbf{A}(i, x)^2}{\sum_{i,j} \mathbf{A}(i, j)^2}$
3. for $i = 1 : c$  [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:, i) = \mathbf{A}(:, j) / \sqrt{cP(j)}$

Note this is a randomized algorithm, same column can be sampled more than once
Computing U

- Let $\mathbf{W}$ be the “intersection” of sampled columns $\mathbf{C}$ and rows $\mathbf{R}$
  - Let SVD of $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}^T$
- Then: $\mathbf{U} = \mathbf{W}^+ = \mathbf{Y} \mathbf{Z}^+ \mathbf{X}^T$
  - $\mathbf{Z}^+$: reciprocals of non-zero singular values: $\mathbf{Z}_{ii}^+ = 1/ \mathbf{Z}_{ii}$
  - $\mathbf{W}^+$ is the “pseudoinverse”

Why pseudoinverse works?
$\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}$ then $\mathbf{W}^{-1} = \mathbf{X}^{-1} \mathbf{Z}^{-1} \mathbf{Y}^{-1}$
Due to orthonormality $\mathbf{X}^{-1} = \mathbf{X}^T$ and $\mathbf{Y}^{-1} = \mathbf{Y}^T$
Since $\mathbf{Z}$ is diagonal $\mathbf{Z}^{-1} = 1/\mathbf{Z}_{ii}$
Thus, if $\mathbf{W}$ is nonsingular, pseudoinverse is the true inverse
CUR: Provably good approx. to SVD

For example:

- Select $c = O\left(\frac{k \log k}{\epsilon^2}\right)$ columns of $A$ using ColumnSelect algorithm.
- Select $r = O\left(\frac{k \log k}{\epsilon^2}\right)$ rows of $A$ using ColumnSelect algorithm.
- Set $U = W^+$.

Then: $\|A - CUR\|_F \leq (2 + \epsilon) \|A - A_k\|_F$ with probability 98%

In practice:
Pick 4k cols/rows for a “rank-k” approximation.
CUR: Pros & Cons

+ Easy interpretation
  • Since the basis vectors are actual columns and rows

+ Sparse basis
  • Since the basis vectors are actual columns and rows

- Duplicate columns and rows
  • Columns of large norms will be sampled many times
Solution

- If we want to get rid of the duplicates:
  - Throw them away
  - Scale (multiply) the columns/rows by the square root of the number of duplicates

A → Rd → Rs
   \[\text{Construct a small U}\]

\[\text{Cd}\]
**SVD vs. CUR**

**SVD:** \[ A = U \Sigma V^T \]

- Huge but sparse
- Big and dense

**CUR:** \[ A = CUR \]

- Dense but small
- Huge but sparse
- Big but sparse
SVD vs. CUR: Simple Experiment

- **DBLP bibliographic data**
  - Author-to-conference big sparse matrix
  - $A_{ij}$: Number of papers published by author $i$ at conference $j$
  - 428K authors (rows), 3659 conferences (columns)
    - Very sparse

- **Want to reduce dimensionality**
  - How much time does it take?
  - What is the reconstruction error?
  - How much space do we need?
Results: DBLP - big sparse matrix

- **Accuracy:**
  - 1 – relative sum squared errors
- **Space ratio:**
  - #output matrix entries / #input matrix entries
- **CPU time**

Sun, Faloutsos: Less is More: Compact Matrix Decomposition for Large Sparse Graphs, SDM ’07.
What we learned today

• Baselines and evaluation metrics

• Dimension reduction (or feature reduction)
  – Principal component analysis (PCA)
  – Singular value decomposition (SVD)
Homework

• Readings
  – Mining of Massive Datasets, chapter 11