Typical Machine Learning Workflow:

- Collect training data
- Convert raw input/feature data (e.g. images, review text) into a numerical vector $x$
- Postulate predictive model with unknown parameters $\theta$
- Use training data to “learn” a good choice of model parameters $\theta$. Usually formulated as an optimisation problem, minimising empirical loss/risk function $J(\theta)$.
- Now use this trained model to make predictions ...

Key is to reflect upon the factors affecting these choices and use critical judgement - there is no set “recipe”.

Assignment 3
Recommender Systems
# Recommender Systems

Example: users rate books they have read from 0-5.

<table>
<thead>
<tr>
<th>Book</th>
<th>Alice(1)</th>
<th>Bob(2)</th>
<th>Carol(3)</th>
<th>Dave(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning for Dummies(1)</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hands-On Machine Learning(2)</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>Deep Learning(3)</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>A Kitten Called Holly(4)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>?</td>
</tr>
<tr>
<td>Kittens 2018 Calendar(5)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Notation:
- $n$ number of users, $n = 4$
- $m$ number of items, $m = 5$
- $d$ number of features
- $R_{uv}$ rating given by user $u$ to item $v$, $R_{11} = 5$
- $\delta_{uv} = 1$ if item $v$ rated by user $u$, 0 otherwise, $\delta_{11} = 1$, $\delta_{12} = 0$
### Content-Based Recommendations

<table>
<thead>
<tr>
<th>Book</th>
<th>Alice(1)</th>
<th>Bob(2)</th>
<th>Carol(3)</th>
<th>Dave(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning for Dummies(1)</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hands-On Machine Learning(2)</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>Deep Learning(3)</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>A Kitten Called Holly(4)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>?</td>
</tr>
<tr>
<td>Kittens 2018 Calendar(5)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

- Associate a feature vector $x^{(v)}$ with $v$'th book, e.g. $x^{(1)} = [1, 0]^T$, $x^{(4)} = [0, 1]^T$ (number of features $d = 2$)
- For each user $u$ learn parameter vector $\theta^{(u)}$, e.g. $\theta^{(1)} = [5, 0]^T$, $\theta^{(3)} = [0, 5]^T$
- Predicted rating by user $u$ of item $v$ is $(\theta^{(u)})^T x$, e.g. rating by user 1 of item 1 its $[5, 0] \times [1, 0]^T = 5$
Content-Based Recommendations

• We are given a feature vector $x^{(v)}$ for $v$'th item/book
• Training data: a set of ratings $\{R_{uv}\}$ by users of a subset of the items (each user might only rate a few items)
• Hypothesis: predicted rating by user $u$ of item $v$ is:
  \[ h_{\theta(u)}(x^{(v)}) = (\theta(u))^T x^{(v)} \]
• Parameters: $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}$
• Cost function:
  \[ J(\theta^{(1)}, \ldots, \theta^{(n)}) = \sum_{u=1}^{n} \sum_{v=1}^{m} \delta_{uv} (R_{uv} - (\theta^{(u)})^T x^{(v)})^2 + \lambda \sum_{u=1}^{n} (\theta^{(u)})^T \theta^{(u)} \]
• Select $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}$ to minimise this cost function. This requires solving a least squares problem: use gradient descent or closed-form solution.
User-Based Recommendations

To predict rating $R_{uv}$ of item $v$ by user $u_0$ use a $k$-nearest neighbours approach:

- Find the $k$ users who are “closest” to user $u_0$, call this set $U$. How to measure who is closest?
  - Euclidean distance $d(u, u_0) = \sqrt{\sum_{v: \delta_{uv}=1, \delta_{u_0v}=1} (R_{uv} - R_{u_0v})^2}$
  - Cosine distance $d(u, u_0) = \frac{\sum_{v: \delta_{uv}, \delta_{u_0v}=1} R_{uv} \times R_{u_0v}}{\sqrt{\sum_{v: \delta_{uv}, \delta_{u_0v}=1} R_{uv}^2} \sqrt{\sum_{v: \delta_{uv}, \delta_{u_0v}=1} R_{u_0v}^2}}$

- Predicted $R_{u_0v}$ is weighted average of the rating of item $v$ by these users e.g.

$$R_{u_0v} = \frac{\sum_{u \in U} K(u, u_0) R_{uv}}{\sum_u K(u, u_0)}$$

where $K(u, u_0) = 1/(1 + d(u, u_0))$ is a weight that decreases with the distance between users $u$ and $u_0$.

Observe that when ratings data is sparse both the user- and the content-based approaches run into difficulties.
Collaborative Filtering

<table>
<thead>
<tr>
<th>Book</th>
<th>Alice(1)</th>
<th>Bob(2)</th>
<th>Carol(3)</th>
<th>Dave(4)</th>
<th>$x^{(1)}$ = [ML kittens]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning for Dummies(1)</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hands-On Machine Learning(2)</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>Deep Learning(3)</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>A Kitten Called Holly(4)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Kittens 2018 Calendar(5)</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

- Associate a feature vector $x^{(v)}$ with $v$'th book. But what if we don’t know $x^{(v)}$?

- Suppose we know $\theta^{(1)} = [5, 0]^T$, $\theta^{(3)} = [0, 5]^T$, then

  $[5, 0]^T x^{(1)} = 5$, $[5, 0]^T x^{(3)} = 5$, $[5, 0]^T x^{(4)} = 0$

  $[0, 5]^T x^{(1)} = 0$, $[0, 5]^T x^{(4)} = 5$

  which is satisfied by $x^{(1)} = [10]$, $x^{(3)} = [10]$, $x^{(4)} = [01]$
Collaborative Filtering

- Given $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}$, select $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ to minimise
  \[
  \sum_{u=1}^{n} \sum_{v=1}^{m} \delta_{uv} (R_{uv} - (\theta^{(u)})^T x^{(v)})^2 + \lambda \sum_{v=1}^{m} (x^{(v)})^T x^{(v)}
  \]

- Define
  \[
  J(x^{(1)}, \ldots, x^{(m)}) = \sum_{u=1}^{n} \sum_{v=1}^{m} \delta_{uv} (R_{uv} - (\theta^{(u)})^T x^{(v)})^2 + \lambda \sum_{v=1}^{m} (x^{(v)})^T x^{(v)} + \lambda \sum_{u=1}^{n} (\theta^{(u)})^T \theta^{(u)}
  \]

- Repeat:
  - Given $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}$, select $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ to minimise $J$
  - Given $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$, select $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}$ to minimise $J$

- Each update requires solving a least squares problem: use gradient descent or closed-form solution. This is called the \textbf{alternating least-squares} algorithm

- Recommendation: predicted rating by user $u$ of item $v$ is $(\theta^{(u)})^T x^{(v)}$
Collaborative Filtering: Matrix Completion

Another way to think about the same thing ...

- Observe ratings $R_{uv}$ by user $u$ for item $v$. Gather these into ratings matrix $R$. We want to predict the missing entries in $R$. This is 'matrix completion' task.

- To proceed, assume $R$ is low rank $d \ll n, m$ ....

$$R = \begin{bmatrix} 1 \cdots 1 \\ \vdots \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = UTV$$

- Hypothesis: $R = U^T V$, but the elements of $U$ and $V$ are unknown.

- Cost Function: $\frac{1}{m} \sum_{u,v} (R_{uv} - (U^T V)_{uv})^2 + \lambda U^T U + \lambda V^T V$
Some Issues ...

• Data sparsity, especially a problem with content/user-based methods:
  • Content-based: not enough ratings to estimate $\theta^{(u)}$ for user $u$
  • User-based: or to find nearest neighbours with enough ratings

• Cold-start problem:
  • What to recommend to a new user (who hasn’t rated anything yet)?
  • How to recommend new items (which have no ratings yet)?
Some Issues ...

‘Popularity bias: hard to recommend to someone with unique tastes”

- Good quality data is always a key issue. Even with lots of data our model doesn’t generalise well i.e. doesn’t predict well for data outside the training set.

- But what is the intrinsic noise when making predictions anyway? E.g. For Netflix data set the state of the art is RMSE of about 0.9. Ratings are concentrated between 3 and 5. So $4 \pm 0.9$ covers almost the whole range.
Issues

- Shilling attacks/adversarial data
  - Create costly barrier to keep bots etc out e.g. booking.com requires paying for a room in hotel before a review can be submitted.
  - Create barrier by building reputation over time e.g. stackoverflow

- And then there’s the question of privacy ...
  ... US, Europe and Asia have very different privacy regulations. Europe is introducing the GDPR next May.
  - As access control (couched as “consent”) ...
  - Adding noise/perturbing the data (k-anonymity, differential privacy etc). Privacy comes at the cost of poorer performance.
  - Hiding in the crowd?
Privacy by Design: Personalised Recommendations

![Diagram showing conventional and OpenNym recommender systems](image)

User accesses system via an account used only by them, so id and user strongly linked.

Existing Recommender System

User Client App

(user id, item, rating)

recommendation

Existing Recommender System

User Client App

(group id, item, rating)

recommendation

User accesses system via an account shared with other users having similar interests.

OpenNym Identities

Privacy by Design: Personalised Recommendations
Summary of the RMSE performance using validation sets from [1]².

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BMF</th>
<th>ALSWR</th>
<th>SVD++</th>
<th>SGD</th>
<th>Bias SVD</th>
<th>BLC local</th>
<th>(nyms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jester</td>
<td>4.33</td>
<td>5.64</td>
<td>5.54</td>
<td>5.72</td>
<td>5.82</td>
<td><strong>4.20</strong></td>
<td>64</td>
</tr>
<tr>
<td>Movielens</td>
<td>0.85</td>
<td>1.51</td>
<td>1.42</td>
<td>1.24</td>
<td>1.23</td>
<td><strong>0.83</strong></td>
<td>26</td>
</tr>
<tr>
<td>Dating</td>
<td>1.93</td>
<td>4.72</td>
<td>4.68</td>
<td>5.17</td>
<td>3.96</td>
<td><strong>1.88</strong></td>
<td>14</td>
</tr>
<tr>
<td>Books</td>
<td>1.94</td>
<td>4.71</td>
<td>4.73</td>
<td>5.18</td>
<td>3.95</td>
<td><strong>1.87</strong></td>
<td>1</td>
</tr>
<tr>
<td>Netflix</td>
<td>0.95</td>
<td>1.56</td>
<td>1.54</td>
<td>1.29</td>
<td>1.38</td>
<td><strong>0.97</strong></td>
<td>128</td>
</tr>
</tbody>
</table>