Beyond link prediction and collaborative filtering:
Learning to predict affinity

Charles Elkan
University of California, San Diego

August 20, 2011
Outline

1. Introduction: Seven related prediction tasks
2. The LFL method
3. Link prediction in networks
4. Bilinear regression to learn affinity
5. Discussion
Link prediction

- Given current friendship edges, predict future edges.

Application: Facebook.

Popular method: compute scores from graph topology.
Collaborative filtering

- Given ratings of movies by users, predict other ratings.

- Application: Netflix.

- Popular method: matrix factorization.
Suggesting citations

- Each author has referenced certain papers. Which other papers should s/he read?


- Method: specialized graphical model.
Gene-protein networks

- In biology, experiments show which regulatory proteins control which genes.

- Application: Energy independence :-)  
- Popular method: support vector machines (SVMs).
Item response theory

- Given answers by students to exam questions, predict performance on other questions.

- Applications: Adaptive testing, diagnosis of skills.

- Popular method: latent trait models.
Compatibility prediction

- Given questionnaire answers, predict successful dates.

- Application: eHarmony.

- Popular method: learn a Mahalanobis (transformed Euclidean) distance metric.
Predicting behavior of shoppers

- A customer’s actions include \{ look at product, put in cart, finish purchase, write review, return for refund \}.

- Application: Amazon.

- New method: LFL (latent factor log linear model).
Three federal judges vote on each appeals case. How would other judges have voted?
Dyadic prediction in general

- Given labels for some pairs of items (some dyads), predict labels for other pairs.

  ![Dyadic prediction diagram](image)

- Popular method: Depends on research community!
Dyadic prediction formally

- **Training set** \( ((r_i, c_i), y_i) \in \mathcal{R} \times \mathcal{C} \times \mathcal{Y} \) for \( i = 1 \) to \( i = n \).
  - \((r_i, c_i)\) is a dyad, \(y_i\) is a label.

- **Output**: Function \( f : \mathcal{R} \times \mathcal{C} \rightarrow \mathcal{Y} \)
  - Often, but not necessarily, transductive.

- Flexibility in the nature of dyads and labels:
  - \(r_i, c_i\) can be from same or different sets, with or without unique identifiers, with or without feature vectors.
  - \(y_i\) can be unordered, ordered, or real-valued.

- For simplicity, talk about users, movies and ratings.
Latent feature models

- Associate latent feature values with each user and movie.
- Each rating is the dot-product of corresponding latent vectors.
- Learn the most predictive vector for each user and movie.

- Latent features play a similar role to explicit features.
- Computationally, learning does SVD (singular value decomposition) with missing data.
What’s new

- Using all available information.
- Inferring good models from unbalanced data.
- Predicting well-calibrated probabilities.
- Scaling up.
- Unifying disparate problems in a single framework.
The bigger picture

- Solve a **predictive** problem.
  - Contrast: Non-predictive task, e.g. community detection.

- Maximize objective defined by an **application**, e.g. AUC.
  - Contrast: Algorithm but no goal function, e.g. betweenness.

- Learn from **all** available data.
  - Contrast: Use only graph structure, e.g. commute time.

- Allow **hubs**, **partial** memberships, **overlapping** groups, etc.
  - Contrast: Ensembles of communities, block models.

- Make training time **linear** in number of edges.
  - Contrast: MCMC, betweenness, SVD.

- Compare accuracy to **best** current results.
  - Contrast: Compare only to classic methods.
Competing approaches from various communities:

- Collaborative filtering: MMMF [Rennie and Srebro, 2005], BPMF [Salakhutdinov and Mnih, 2008], ...
- Link prediction: unsupervised scores [Adamic and Adar, 2001], IBP [Miller et al., 2009], MMSB [Airoldi et al., 2008] ...
- Statistical relational learning

New methods are better in accuracy, scalability, wide applicability. [Menon and Elkan, 2010a], [Menon and Elkan, 2010b], [Menon and Elkan, 2011]
Outline

1. Introduction: Seven related prediction tasks
2. The LFL method
3. Link prediction in networks
4. Bilinear regression to learn affinity
5. Discussion
Desiderata for dyadic prediction

• Predictions are pointless unless used to make decisions.
  ▶ Need probabilities of ratings e.g. $p(5 \text{ stars}|\text{user, movie})$

• What if labels are discrete?
  ▶ Link types may be $\{\text{friend, colleague, family}\}$
  ▶ For Amazon, labels may be $\{\text{viewed, purchased, returned}\}$

• What if a user has no ratings, but has side-information?
  ▶ Combine information from latent and explicit feature vectors.

• Address these issues within the log-linear framework.
The log-linear framework

- A log-linear model for inputs $x \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ assumes
  \[ p(y|x; w) \propto \exp \left( \sum_{i=1}^{n} w_i f_i(x, y) \right) \]

- Predefined feature functions $f_i : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- Trained weight vector $w$.

- Useful general foundation for predictive models:
  - Models probabilities of labels given an example
  - Purely discriminative: no attempt to model $x$
  - Labels can be nominal and/or have structure
  - Combines multiple sources of information correctly.
A first log-linear model for dyadic prediction

- For dyadic prediction, each example $x$ is a dyad $(r, c)$.
- Feature functions must depend on both examples and labels.
- Simplest choice:

$$ f_{r'c'y'}((r, c), y) = 1[r = r', c = c', y = y'] . $$

- Conceptually, re-arrange $w$ into a matrix $W^y$ for each label $y$: 

$$ p(y|(r, c); w) \propto \exp(W^y_{rc}) . $$
Factorizing interaction weights

- **Problem:** \( 1[ r = r', c = c', y = y'] \) is too specific to individual \((r', c')\) pairs.

- **Solution:** Factorize the \( W_y \) matrices. Write \( W_y = A^T B \) so

\[
W_{rc}^y = (\alpha^y_r)^T \beta^y_c = \sum_{k=1}^{K} \alpha^y_{rk} \beta^y_{ck}
\]

For each \( y \), each user and movie has a vector of values representing characteristics that predict \( y \).

- In practice, a single vector of movie characteristics suffices:
  \( \beta^y_c = \beta_c \)

- The characteristics predicting that a user will rate 1 star versus 5 stars are different.
Incorporating side-information

- If a dyad \((r, c)\) has a vector \(s_{rc} \in \mathbb{R}^d\) of side-information, define
  \[
p(y|(r, c); w) \propto \exp((\alpha^y_r)^T \beta^y_c + (v^y)^T s_{rc}).
  \]
- Multinomial logistic regression with \(s_{rc}\) as feature vector.
What if features are only per-user $u_r$ or per-movie $m_c$?

Naïve solution: Define $s_{rc} = [u_r \ m_c]$.

- But then all users have the same rankings of movies.

Better: Apply bilinear model to user and movie features

$$p(y|(r, c); w) \propto \exp((\alpha_r^y)^T \beta_c^y + u_r^T V^y m_c).$$

The matrix $V^y$ consists of weights on cross-product features.
The LFL model: definition

- Resulting model with latent and explicit features:

\[ p(y|(r, c); w) \propto \exp((\alpha^y_r)^T \beta^y_c + (v^y)^T s_{rc} + u_r^T V^y m_c) \]

- \(\alpha^y_r\) and \(\beta^y_c\) are latent feature vectors in \(\mathbb{R}^K\).
  - \(K\) is number of latent features

- Practical details:
  - Fix a base class for identifiability.
  - Intercept terms for each user and movie are important.
  - Use \(L_2\) regularization.
  - Train with stochastic gradient descent (SGD).
Unordered versus numerical labels

- For unordered ratings, predict the most probable, and train to optimize log likelihood.

- Not desirable for numerical ratings:
  - Difference between 1 and 5 \(\neq\) difference between 4 and 5

- Better: Predict

\[
\mathbb{E}[y] = \sum_{y=1}^{|Y|} y \cdot p(y|(r, c); w)
\]

and optimize mean squared error \(\text{MSE}\).

  - The expectation \(\mathbb{E}[y]\) is a summary function.
  - A standard latent feature model is limited to one factorization for all rating levels.
Assessing uncertainty

- The variance measures the uncertainty of a prediction.
- For numerical ratings

\[ \mathbb{E}[y^2] - (\mathbb{E}[y])^2 = \sum y^2 \cdot p(y|(r,c); w) - \left( \sum y \cdot p(y|(r,c); w) \right)^2 \]

- Can be combined with business rules, e.g. for gene-protein regulation, if confidence in predicted link < cost threshold then do not run expensive experiment.
Experimental goals

- Show ability to
  - Handle unordered labels for multiclass link prediction
  - Exploit numerical structure of labels for collaborative filtering
  - Incorporate side-information in a cold-start setting.

- Later:
  - More detailed study of link prediction
  - Complementarity of explicit and latent features.
Multiclass link prediction

- The Alyawarra dataset has kinship relations \{\text{brother, sister, father, } \ldots\} between 104 people.

- LFL outperforms Bayesian models, even infinite ones.
  - MMSB, IRM assume interactions set by cluster membership.
  - IBP has binary latent features.

- Bayesian averaging over multiple models does not add power.
Collaborative filtering

- **MovieLens** (6040 users, 3952 movies, 1M ratings of 1-5 stars)
- **EachMovie** (36,656 users, 1628 movies, 2.6M ratings of 1-6 stars)

- LFL model is more general, more accurate, and faster than maximum margin matrix factorization [Rennie and Srebro, 2005].
Measuring uncertainty

- Estimated uncertainty correlates with observed test set errors and average rating of movie. For MovieLens:

<table>
<thead>
<tr>
<th>Lowest variance</th>
<th>Highest variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kazaam</td>
<td>Grateful Dead</td>
</tr>
<tr>
<td>Lawnmower Man 2: Beyond Cyberspace</td>
<td>The Rescuers</td>
</tr>
<tr>
<td>Problem Child 2</td>
<td>Prizzi’s Honor</td>
</tr>
<tr>
<td>Meatballs III</td>
<td>Homeward Bound: The Incredible Journey</td>
</tr>
<tr>
<td>Pokemon the Movie 2000</td>
<td>The Fly</td>
</tr>
</tbody>
</table>
Side-information solves the cold-start problem

- Three scenarios on the 100K MovieLens dataset:
  - **Standard**: No cold-start for users or movies
  - **Cold-start users**: Randomly discard ratings of 50 users
  - **Cold-start users + movies**: Randomly discard ratings of 50 users and ratings for all their test set movies also.
Outline

1. Introduction: Seven related prediction tasks
2. The LFL method
3. Link prediction in networks
4. Bilinear regression to learn affinity
5. Discussion
Link prediction

- **Link prediction**: Given a *partially observed* graph, predict whether or not edges exist for the unknown-status pairs.

Unsupervised (non-learning) scores are classical models:
- e.g. common neighbors, Katz measure, Adamic-Adar.

Technically, **structural** rather than **temporal** link prediction.
Latent feature approach

- Each node’s identity influences its linking behavior.
- Nodes also can have side-information predictive of linking.
  - For author-author linking, side-information can be words in authors’ papers.
- Edges may also possess side-information.
  - For country-country conflict, side-information is geographic distance, trade volume, etc.
- Identity determines latent features.
Latent feature approach

- LFL model for binary link prediction has parameters
  - latent vectors $\alpha_i \in \mathbb{R}^k$ for each node $i$
  - scaling factors $\Lambda \in \mathbb{R}^{k \times k}$ for asymmetric graphs
  - weights $W \in \mathbb{R}^{d \times d}$ for node features
  - weights $v \in \mathbb{R}^{d'}$ for edge features.

- Given node features $x_i$ and edge features $z_{ij}$

$$\hat{G}_{ij} = \sigma(\alpha_i^T \Lambda \alpha_j + x_i^T W x_j + v^T z_{ij})$$

for sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$

- Minimize regularized training loss:

$$\min_{\alpha, \Lambda, W, v} \sum_{(i,j) \in \mathcal{O}} \ell(G_{ij}, \hat{G}_{ij}) + \Omega(\alpha, \Lambda, W, v)$$
Challenge: Class imbalance

- Vast majority of node-pairs do not link with each other.
- AUC (area under ROC curve) is standard performance measure.
- For a random pair of positive and negative examples, AUC is the probability that the positive one has higher score.
  - Not influenced by relative size of positive and negative classes.
- Model trained to maximize accuracy is potentially suboptimal.
  - **Sampling** is popular, but loses information.
  - **Weighting** is merely heuristic.
Optimizing AUC

- Empirical AUC counts concordant pairs
  \[
  \mathcal{A} \propto \sum_{p \in +, q \in -} 1[f_p - f_q > 0]
  \]

- Train latent features to maximize approximation to AUC:
  \[
  \min_{\alpha, \Lambda, W, v} \sum_{(i, j, k) \in \mathcal{D}} \ell(\hat{G}_{ij} - \hat{G}_{ik}, 1) + \Omega(\alpha, \Lambda, W, v)
  \]

  where \(\mathcal{D} = \{(i, j, k) : G_{ij} = 1, G_{ik} = 0\}\).

- With stochastic gradient descent, a fraction of one epoch is enough for convergence.
Experimental comparison

Compare

- latent features versus unsupervised scores
- latent features versus explicit features.

Datasets from applications of link prediction:

- **Computational biology**: Protein-protein interaction network, metabolic interaction network
- **Citation networks**: NIPS authors, condensed matter physicists
- **Social phenomena**: Military conflicts between countries, U.S. electric power grid.
Link prediction datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>$\mathcal{O}^+$</th>
<th>$\mathcal{O}^-$</th>
<th>$+$:−$+$ ratio</th>
<th>Average degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prot-Prot</td>
<td>2617</td>
<td>23710</td>
<td>6,824,979</td>
<td>1:300</td>
<td>9.1</td>
</tr>
<tr>
<td>Metabolic</td>
<td>668</td>
<td>5564</td>
<td>440,660</td>
<td>1:80</td>
<td>8.3</td>
</tr>
<tr>
<td>NIPS</td>
<td>2865</td>
<td>9466</td>
<td>8,198,759</td>
<td>1:866</td>
<td>3.3</td>
</tr>
<tr>
<td>Condmat</td>
<td>14230</td>
<td>2392</td>
<td>429,232</td>
<td>1:179</td>
<td>0.17</td>
</tr>
<tr>
<td>Conflict</td>
<td>130</td>
<td>320</td>
<td>16580</td>
<td>1:52</td>
<td>2.5</td>
</tr>
<tr>
<td>PowerGrid</td>
<td>4941</td>
<td>13188</td>
<td>24,400,293</td>
<td>1:2000</td>
<td>2.7</td>
</tr>
</tbody>
</table>

- Protein-protein interaction data from Noble. Each protein has a 76 dimensional explicit feature vector.
- Metabolic pathway interaction data for *S. cerevisiae* provided in the KEGG/PATHWAY database [ISMB]. Each node has three feature sets: a 157 dimensional vector of phylogenetic information, a 145 dimensional vector of gene expression information, and a 23 dimensional vector of gene location information.
- NIPS: Each node has a 14035 dimensional bag-of-words feature vector, the words used by the author in her publications. LSI reduces the number of features to 100.
- Co-author network of condensed-matter physicists [Newman].
- Military disputes between countries [MID 3.0]. Each node has 3 features: population, GDP and polity. Each dyad has 6 features, e.g. the countries’ geographic distance.
- US electric power grid network [Watts and Strogatz].
Latent features versus unsupervised scores

- Latent features are more predictive of linking behavior.
Learning curves

- Unsupervised scores need many edges to be known.
- Latent features are predictive with fewer known edges.
- For the military conflicts dataset:
Latent features combined with side-information

- Difficult to infer latent structure more predictive than side-information.
- But combining the two can be beneficial:
Outline

1. Introduction: Seven related prediction tasks
2. The LFL method
3. Link prediction in networks
4. Bilinear regression to learn affinity
5. Discussion
What is affinity?

- Affinity may be called similarity, relatedness, compatibility, relevance, appropriateness, suitability, and more.
  - Two OCR images are *similar* if they are versions of the same letter.
  - Two eHarmony members are *compatible* if they were mutually interested in meeting.
  - An advertisement is *relevant* for a query if a user clicks on it.
  - An action is *suitable* for a state if it has high long-term value.

- Affinity can be between items from the same or different spaces.

- Affinity can be binary or real-valued.
The propensity problem

- **Idea:** To predict affinity, train a linear function
  \[
  f(u, v) = w \cdot [u, v].
  \]

- **Flaw:** Ranking of second entities \(v\) is the same regardless of \(u\):
  \[
  f(u, v) = w \cdot [u, v] = w_u \cdot u + w_v \cdot v.
  \]

  The ranking of \(v\) entities is by the dot product \(w_v \cdot v\).
Bilinear representation

- Proposal: Represent affinity of vectors $u$ and $v$ with a function
  \[ f(u, v) = u^T W v \]

  where $W$ is a matrix.

- Different vectors $u$ give different ranking vectors $w(u) = u^T W$. 
Learning $W$

- A training example is $\langle u, v, y \rangle$ where $y$ is a degree of affinity.
- Let $u$ and $v$ have length $m$ and $n$. Then

$$u^T W v = \sum_{i=1}^{m} \sum_{j=1}^{n} (W \circ uv^T)_{ij} = \text{vec}(W) \cdot \text{vec}(uv^T).$$

- Idea: Convert $\langle u, v, y \rangle$ into $\langle \text{vec}(uv^T), y \rangle$.
- Then learn $\text{vec}(W)$ by standard linear regression.
What does $W$ mean?

- Each entry of $uv^T$ is the interaction of a feature of the $u$ entity and a feature of the $v$ entity.
- Labels may be real-valued or binary: $y = 1$ for affinity, $y = 0$ for no affinity.
- Can use regularization, logistic regression, linear SVM, and more.
- Can maximize AUC.
Add a constant 1 to $u$ and $v$ to capture propensities.

If $u$ and $v$ are too short, expand them, e.g. change $u$ to $uu^T$.

If $u$ and/or $v$ is too long, define $W = AB^T$ where $A$ and $B$ are rectangular.

If $W$ is square, define $W = AB^T + D$ where $D$ is diagonal.

But finding the optimal representation $AB^T$ or $AB^T + D$ is not a convex problem.
Affinities versus distances

- Learning affinity is an alternative to learning a distance metric.
- The Mahalanobis metric is \( d(u, v) = \sqrt{(u - v)^T M (u - v)} \) where \( M \) is positive semidefinite.
- Learning affinities is more general.
  - Distance is defined only if \( u \) and \( v \) belong to the same space.
  - In information retrieval, \( u \) can be a query in one language and \( v \) can be a relevant document in a different language.
- Affinity is not always symmetric.
  - Because queries are shorter than documents, the relatedness of queries and documents is not symmetric.
Squared Mahalanobis distance is $d^2(u, v) =$

$$(u - v)^T M (u - v) = \sum_{i=1}^{n} \sum_{j=1}^{n} (M \circ (u - v)(u - v)^T)_{ij}$$

$$= \text{vec}(M) \cdot \text{vec}((u - v)(u - v)^T).$$

So $M$ can be learned by linear regression, like $W$.

The outer product $(u - v)(u - v)^T$ is symmetric, so $M$ is symmetric also.

Existing methods for learning Mahalanobis distance are less efficient.
Experiments with eHarmony data

- The training set has 506,688 labeled pairs involving 274,654 members of eHarmony, with 12.3% positive pairs.
- The test set has 439,161 pairs involving 211,810 people, with 11.9% positive pairs.
- Previously used in [McFee and Lanckriet, 2010].
Visualization

- Positive training pairs from the U.S. and Canada.

Each line segment connects the locations of two individuals in the eHarmony training set who are compatible.
Data representations

- Each user is a vector of length $d = 56$. “Propensity” uses vectors of length $2d + 1$
- “Interaction” uses length $3d + 1$ by adding $u_i v_i$ for $i = 1$ to $d$.
- “Extended interaction” adds nonlinear transformations of components $u_i$ and $v_i$.
- “Bilinear” uses vectors of length $d^2$.
- “Mahalanobis” uses vectors of length $d(d + 1)/2 = 1597$.
- Extended bilinear and Mahalanobis representations use quadratic vectors concatenated with extended interaction vectors.
Experimental details

- Training uses linear regression with an intercept.
- Targets are 0 or 1. Features are z-scored.
- $L_2$ regularization with strength one.
- For comparability, id numbers, latitudes, and longitudes are ignored.
Experimental results

- Training and test AUC for alternative representations.

<table>
<thead>
<tr>
<th>representation</th>
<th>training AUC</th>
<th>test AUC</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLR-MAP</td>
<td>0.624</td>
<td></td>
<td></td>
</tr>
<tr>
<td>propensity</td>
<td>0.6299</td>
<td>0.6354</td>
<td>14</td>
</tr>
<tr>
<td>interaction</td>
<td>0.6410</td>
<td>0.6446</td>
<td>20</td>
</tr>
<tr>
<td>extended interaction</td>
<td>0.6601</td>
<td>0.6639</td>
<td>64</td>
</tr>
<tr>
<td>Mahalanobis</td>
<td>0.6356</td>
<td>0.6076</td>
<td>379</td>
</tr>
<tr>
<td>extended Mahalanobis</td>
<td>0.6794</td>
<td><strong>0.6694</strong></td>
<td>459</td>
</tr>
<tr>
<td>bilinear</td>
<td>0.6589</td>
<td>0.6374</td>
<td>973</td>
</tr>
<tr>
<td>extended bilinear</td>
<td>0.6740</td>
<td>0.6576</td>
<td>1324</td>
</tr>
</tbody>
</table>

- The large test set makes differences statistically significant.
Observations

- Bilinear regression is tractable. Training with a half million examples of expanded length 3000 takes 22 minutes.
- Learning propensity is a strong baseline, with higher accuracy than the best previous method.
- Bilinear affinity gives higher accuracy than Mahalanobis distance.
- A nonlinear extended version of Mahalanobis distance is best overall.
Outline

1. Introduction: Seven related prediction tasks
2. The LFL method
3. Link prediction in networks
4. Bilinear regression to learn affinity
5. Discussion
If time allowed


- Predicting labels for nodes, e.g. who will play Farmville (within network classification, collective classification).
Conclusions

- Many prediction tasks involve pairs of entities: collaborative filtering, friend suggestion, compatibility forecasting, reinforcement learning, and more.

- Edge prediction based on learning latent features is always more accurate than prediction based on any fixed graph-theoretic formula.

- The most successful methods combine latent features with explicit features of nodes and of dyads.


