情報推薦 (Recommendation)

情報推薦 (Recommendation)

ユーザ集合を $U = \{u_1, u_2, \ldots, u_m\}$, アイテム集合を $I = \{i_1, i_2, \ldots i_n\}$ とする. このとき, ユーザ (u_q) にどのアイテムを推薦するかを考える. ユーザが高い評価すると推測されるアイテムを見つける手法. 情報推薦は情報フィルタリングの 1 つ. 内容ベースフィルタリング (Content-based Filtering) と協調フィルタリング (Collaborative Filtering) に大別される.

内容ベースフィルタリング (Content-based Filtering)

ユーザが高い評価をしたアイテムに似たアイテムを推薦する. コサイン類似度 (Cosign Similarity) によって評価する. アイテム (i_j) に関する情報ベクトルを $\mathbf{v}_j$ とする. 利点はアイテムの特徴もとに推薦するため, コールドスタート問題を回避できること. 課題は過去の評価アイテムとベクトルが似ているものを推薦するため, 同じカテゴリのものを推薦しやすいこと.

$$cos(\mathbf{v}_j, \mathbf{v}_h) = \frac{\mathbf{v}_j^\top \mathbf{v}_h}{|\mathbf{v}_j||\mathbf{v}_h|}$$

協調フィルタリング (Collaborative Filtering)

好みが似ている他のユーザを探し, そのユーザが高くしたアイテムを推薦する. (ユーザーベース協調フィルタリング) または, ユーザが過去に好んだアイテムと類似したアイテムを推薦する. (アイテムベース協調フィルタリング) User-item Rating Matrix $R \subseteq \mathbb{R}^{m \times n}$ を利用する. 各要素 $r_{j,h}$ は, ユーザ u_j のアイテム i_h に対する評価を表す. よって列ベクトル I_h はアイテム i_h のユーザによる評価ベクトル, 行ベクトル U_j はユーザ u_j のアイテムに対する評価ベクトルを表す.

類似度

好みが似ている他のユーザや似ているアイテムをどのように探すか. ユーザ u_q と u_j との類似度 $w(u_q, u_j)$ を数値化する. アイテムに対する評価が似ているユーザが似ている. よって U_q と U_j を比較する. コサイン類似度によって評価する.

$$cos(U_q, U_j) = \frac{U_q^\top U_j}{|U_q||U_j|}$$

ピアソン相関類似度 (Pearson Correlation Similarity) によって評価する. コサイン類似度のユーザごとの評価の偏りをユーザー平均で補正する.

$$p(U_q, U_j) = \frac{(U_q - \bar{U_q})^\top (U_j - \bar{U_j})}{|(U_q - \bar{U_q})||(U_j - \bar{U_j})|}$$

アイテムについても同様.

ユーザベース協調フィルタリング

ユーザ u_q のアイテム i_h に対する評価 $p_{q,h}$ を推測する. ユーザの評価の平均よりもどれだけ大きくなるかを考える. 他のユーザの評価の差を利用する. ユーザ間の類似度による加重平均によって計算する. すべてを計算するのは時間がかかるため, 特性が異なるユーザの情報は不必要として類似度の値が大きいユーザのみで計算することが多い.

課題はアイテム数に応じて計算量が増えること, ユーザの評価において共通のアイテムが少ない場合に精度が落ちること, ユーザの評価は常に変化していること.

$$p_{q,h} = \bar{r_q} + k \sum_{j=1}^{m-1} w(u_q, u_j)(r_{j,h} - \bar{r_j})$$

$w(u_q, u_j)$ は類似度による重み, k はハイパーパラメータ (加重平均なのでおそらく $k = \frac{1}{\sum_{j=1}^{m-1} w(u_q, u_j)}$), $\bar{r_j}$ はユーザ u_j の評価の平均とする.

アイテムベース協調フィルタリング

ユーザ u_q のアイテム i_h に対する評価 $p_{q,h}$ を推測する. アイテムの評価の平均よりもどれだけ大きくなるかを考える.

$$p_{q,h} = \bar{r_h} + k \sum_{k=1}^{n-1} w(i_q, i_k)(r_{q,k} - \bar{r_k})$$

スロープワン (Slope One)

ユーザ u_j の異なるアイテム i_h, i_k の評価の偏差を利用して, ユーザ u_q のアイテム t_h に対する評価を推測する.

$$dev^j_{h,k} = r_{j,h} - r_{j,k}$$

ユーザ u_j の評価に基づけば, ユーザ u_q のアイテム t_h に対する評価は以下のように表せる.

$$p_{q,h} = r_{q,k} + dev^j_{h,k}$$

これをすべてのユーザとアイテムについて拡張する. 偏差の平均を求める. $U(h,k)$ は i_h, i_k の両方を評価したユーザの集合とする.

$$dev_{h,k} = \frac{1}{|U(h,k)|} \sum_{u_j \in U(h,k)} dev^j_{h,k}$$

すべてのアイテムについて, 求めた評価予測の平均.

$$p_{q,h} = \frac{1}{|I_q|} \sum_{i_k \in I_q} (r_{q,h}, dev_{h,k})$$

評価した人数が多い方が信頼性が高いことから, 評価予測に重みをつけ加重平均とする. $w_{h,k}$ は i_h, i_k の両方を評価したユーザ数 ($|U(h,k)|$) とする.

$$p_{q,h} = \frac{\sum_{i_k \in I_q} w_{h,k}(r_{q,h}, dev_{h,k})}{\sum_{i_k \in I_q} w_{h,k}}$$

問題点

問題点としてスケーラビリティ (Scalability), コールドスタート (Cold Start), スパース性 (Sparsisty) がある. スケーラビリティはアイテム数, ユーザ数が多いと計算時間がかかる問題. コールドスタートは新しいユーザやアイテムはごくわずか評価しかない (されない) ので, データが不足してしまう問題. 内容ベースフィルタリングによって回避できる. スパース性はアイテム数が多いと, 評価されていないアイテムが増えることでデータに空 (N/A) がふえること. 評価行列を分解することで回避できる.

$$\mathbf{R}^{m\times n} \to \mathbf{U}^{m\times k}\mathbf{I}^{k\times m} = \hat{\mathbf{R}}$$

m, n より十分に小さい k で, $|\mathbf{R} - \hat{\mathbf{R}}|$ を最小とする k を求める.

評価

精度 (Precision) 推薦されたアイテムの中で, どれだけ正しいアイテムが含まれているか. 再現率 (Recall) 推薦すべきアイテムをどれだけ正確に推薦したか. F 値 (F1 Score) 精度と再現率のバランスを評価.