Week7 Learning Reflection

Name: Xinyu Chang
Time: 2022.05.09-05.15
Class: CSE 416 Intro. Machine Learning
Lectures: 13 and 14

Summary

1. In practice, SVD might be a useful way to compute eigenvectors during the PCA steps.
2. Precision and Recall tend to show a negative relationship.
3. Generally, for a given precision, we want recall to be as large as possible(vice versa)
    a. one metric: area under the curve (AUC)
    b. another metric: Set desired recall and maximize precision (precision at k)

Concepts

(Lecture 13)

Pre-class material

- Recommender Systems
    - recommends items to a user based on what we think will be the most "useful" for the user.
- Recommend System Challenges
    - explicit - user tells us what she likes
    - implicit - we try to infer what she likes from usage data
    - cold start problem.
        - solution: side information
    - top-k vs. diverse recommendations
        - top-k: redundant
        - diverse: users are multi-faceted and we want to hedge our bets
- Popularity Model
    - Simplest Approach: Recommend whatever is popular
    - Limitation: No personalization; Feedback loops
- Classification Model
    - Train a classifier to learn whether or not someone will like an item
    - Pros:
        - personalized;
        - features can capture contexts;
        - can even handle limited user history
    - Cons:
        - features might not be available or hard to work with;
        - Often doesn't perform well in practice when compared to more advanced techniques like collaborative filtering.
        - can still lead to feedback loops
- Feedback loops

In-class Material

- Dimensionality reduction: the task of representing the data with a fewer number of dimensions, while keeping meaningful relations between data
  - example: embedding pictures.
    - embed high dimensional data in low dimensions to visualize the data.
    - Goal: Similar images should be near each other
  - Easier Learning: fewer parameters, no curse of dimensionality
  - Hard to visualize more than 3D
  - Discover intrinsic dimensionality of the data
    - high dimensional data can sometimes be truly lower dimensional.
- Projection
  - Linear projection
    - Project data into 1 dimension along a line
    - Could do something like feature importance and find the subset of features with the most meaningful information
    - A popular approach is to create new features that are combinations of existing features.
    - Benefit: We can do this in the unsupervised setting when we only know x and not y.
- Principal Component Analysis (PCA)
  - Idea: Use a linear projection from d-dimensional data to k-dimensional data
  - Algorithm
    - Input data: An n*d data matrix X: each row is an example
    - 1. Recenter Data: Subtract mean from each row

      $$X_c <- \ X - \bar{X}[1:d]$$

    - 2. Compute spread/orientation: Compute covariance matrix $\sum$

      $$\sum[t,s] \ = \ \frac{1}{n}\sum_{i=1}^{n} x_{c,i}[t]x_{c,i}[s]$$

    - 3. Find basis for orientation: Compute eigenvectors of $\sum$
      - Select k eigenvectors $u_1, \dots, u_k$ with largest eigenvalues
    - 4. Project Data: Project data onto principal

      $$z_i[1] \ = \ u_1^T x_{c,i}$$

      …

      $$z_i[k] \ = \ u_k^T x_{c,i}$$

- Choose the projection that minimizes reconstruction error
    - Idea: The information lost if you were to undo the projection
- Reconstruction error
  - Reconstruction: Reconstruct original data only knowing the projection

$$\hat{x}_i[1:d] = \bar{X}[1:d] + \sum_{j=1}^{k} z_i[j]\, u_j$$

  - error: $||\hat{x}_i[1:d] - x_i||_2^2$
- How to choose the line direction that minimizes reconstruction error
- Geometric understanding of PCA
- Eigenvectors / Eigenfaces
  - Note: We don't particularly care about the math/algorithms to find eigenvectors. Covered in linear algebra classes :)
- When PCA doesn't work
  - PCA assumes there is a low dimensional linear subspace that represents the data sell.
    - May want to look into non-linear dimensionality reduction
      - Manifold learning
      - Popular: SDD Maps, Isomap, LLE, t-SNE
  - Covariance matrix can be very large with high-dimensional data
  - this means finding the principal components will be slow
  - In practice, can use Singular Value Decomposition
    - can be used to find the k eigenvectors with largest eigenvalue
    - very fast implementations

Section material

- Matrix Completion
  - is an impossible task without some assumptions on data.
  - $\hat{Rating}(u, v) = \hat{L}_u \cdot \hat{R}_v$
  - Quality metric
  - $\hat{L}, \hat{R} = argmin_{L,R} \sum_{u,v:\ineq u} (L_u * R_v - r_{uv})^2$
- Coordinate Descent
  - Issue: No unique solution for determining L and R
  - Use coordinate descent to optimize for one coordinate at a time
  - If the system underdetermined, use regularization
- Cold Start and Blended Model
  - Issue: we have a new user and want to give personalized recommendations based on their preferences

- Solution: Blended Model/Featurized Matrix Factorization

(Lecture 14)

Pre-class material

- Co-occurrence matrix
    - Idea: People who bought this, also bought…
    - $C \in R^{m\times m}$ $(m \text{ is the number of items})$
    - $C_{ij} = \# \text{ of users who bought both item } i \text{ and } j$
    - Recommand people things with largest counts
    - Issues:
        - Popular items might drown out other useful items.
    - Solutions
        - Normalization: The count matrix C needs to normalized. Normalize the counts by using the Jaccard similarity.
        - Could also use something like Cosine similarity, but Jaccard is popular.
    - Issue:
        - What if I know the user has bought diapers and milk?
    - Solution:
        - Take the average similarity between items they have bought.
        - Or can put different weight,
        - then find the item with the highest average score.
    - Pros
        - It personalized to user
    - Cons
        - Does not utilize context (time of day), user features (age), product features; scalability (m^2); cold start problem.
- Jaccard similarity- capture how often the co-occurrence appear
    - $S_{ij} = \frac{\# \text{ purchased } i \text{ and } j}{\# \text{ purchased } i \text{ or } j} = \frac{C_{ij}}{T_i + T_j - C_{ij}}$ $\text{where } T_i \text{ the number of time people buy } i \text{ in total}$
- Matrix factorization
    - Need Assumptions: e.g. Assume there are k types of movies, which users have various interests in.
    - User factors
        - eg. how much the user likes each of the categories.
    - Item factors
        - eg. how much the movie belongs to each of the category.

In-class Material

- Matrix factorization quality metric

- Find L and R that when multiplied, achieve predicted ratings that are close to the values hat we have data for.
- $\hat{L}, \hat{R} = argmin_{L,R} \sum_{u,v:r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$
- $L_u \cdot R_v - predicted\, rating;\ r_{uv} - actual\, rating$

- Coordinate descent
  - i.e. Alternating Least Squares
  - implement CD to find $\hat{L}\ and\ \hat{R}$
  - Gradient Descent is not used much in practice to optimize this.
  - Goal: Minimize some function $g(w) = g(w_0, w_1, ..., w_D)$
  - Instead of finding optima for all coordinates, do it for one coordinate at a time,
  - Guaranteed to find an optimal solution under some constraints
  - Algorithm:
    - Initialize $\hat{w} = 0\ (or\ smartly)$
    - while not converged:
      - pick a coordinate j
      - $\hat{w}_j = argmin_w g(\hat{w}_0, \ldots, \hat{w}_{j-1}, w, \hat{w}_{j+1}, \ldots, \hat{w}_D)$
      - ↑ here, we keep all other w's except w_j as constant.
- Coordinate descent for matrix factorization
  - Fix movie factors R and optimize for L_u
    - Independent least squares for each user (n optimizations)
    - $V_u = all\, movies\, user\, u\, has\, rated$
    - $argmin_{L_!, \ldots, L_n} \sum_{u,v:\, r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$
    - $= argmin_{L_!, \ldots, L_n} \sum_u \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$
    - changing of the factors for one user, only will affect the predictions of that one user.
      - Solve for each user u:
      - $\hat{L}_u = argmin_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$
    - Looks like linear regression:
      - $argmin_w \sum_{i=1}^{n} (w^T h(x_i) - y_i)^2$
      - use Gradient Descent: with k coefficients for one user.
  - Fix user factors, and optimize for movie factors separately
    - Independent least squares for each movie (m optimizations)
    - $\hat{R}_u = argmin_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2$
  - Repeatedly do these steps until converge to local optima

    - Underdetermined -> use regularization.
    - nk+mk parameters we will have to learn.
- Topics
    - The features found by matrix factorization don't always correspond to something meaningful
    - The exact values are meaningless but the directions mean something
- Blending models: Learn a model to supplement the matrix factorization model
    - Cold Start problem exists.
        - create a feature vector for each movie
        - Define weights on these features for all users, global.
            - $w_G \in R^d$
        - Fit linear model
            - $\hat{r}_{uv} = w_G^T h(v)$
            - $\hat{w}_G = argmin_w \sum_{r_{u,v}:r_{uv} \neq ?} (w^T h(v) - r_{uv})^2 + \lambda||w||$
        - Prob: feedback loop exists. No general solution, needs to depend on context. Relatively new area need to explore.
    - Featurized matrix factorization
        - Feature-based approach
            - Feature representation of user and movie fixed
            - Can address cold sart problem
        - Matrix factorization approach
            - Suffers from cold start problem
            - User and Movie features are learned from data
        - A unified model
            - $\hat{r}_{uv} = f(\hat{L}_u \cdot \hat{R}_v, (w_G + w_u)^T h(u, v))$
- Evaluating recommender systems - precision/recall
    - We want to look at our set of recommendations and ask:
        - How many of our recommendations did the user like? Precision
        - How many of the items that the user liked did we recommend? Recall
    - Precision
        - $\frac{\#\ liked\ and\ shown}{\#\ shown}$
    - Recall
        - $\frac{\#\ liked\ and\ shown}{\#\ liked}$

Uncertainties

1. Why do we want to find the PCA direction that will maximize the variance along with it?
    a. More variance, so that it will show the differences clearer.