Learning Reflection Week 7

Created	@August 12, 2024 2:48 AM

Summary

This week, we learned about recommender systems, which are used to recommend movies, songs, and news articles to users. There are many algorithms that do this, including by popularity, NN, item-item, user-user, and matrix factorization. Additionally, we discussed dimensionality reduction and specifically how PCA (principal component analysis) can help us visualize the data better and deploy more computationally efficient models. Lastly, we compared all of the models to each other and settled with a hybrid model that is kind of like an ensemble method.

Uncertainties

- How does coordinate descent differ from gradient descent and gradient ascent?
- How would matrix factorization be used in unsupervised learning scenarios?
- How exactly does using a ridge penalty help reduce the number of solutions that minimize the MSE on the quality metric?
- Would using an ensemble model with all of the techniques here weighted to different amounts be a good solution to boost accuracy and display diverse recommendations?

Concepts

Recommender System Intro

- problems
    - interests changer over time

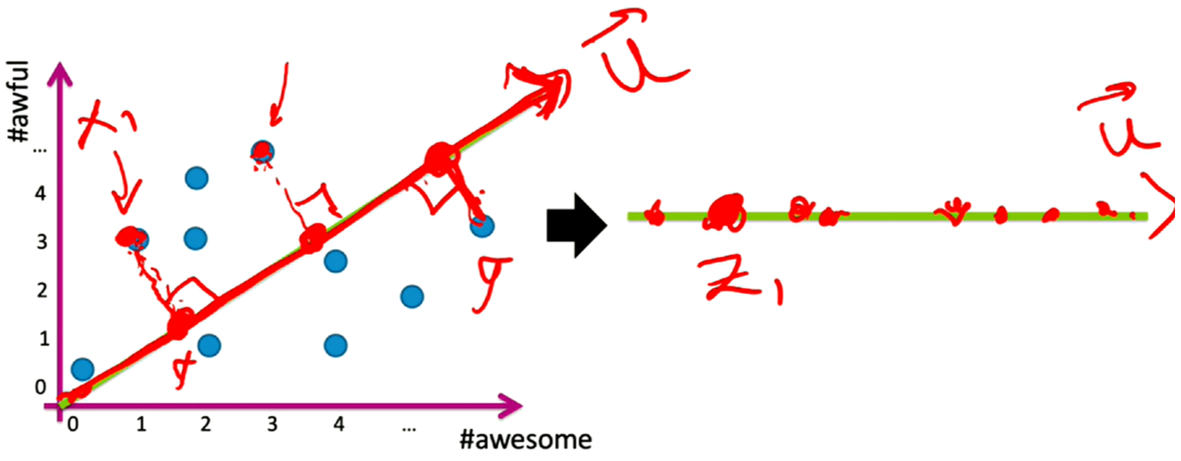
    - you don’t want recommendations to be very close, e.g. recommending all 5 movies of Dune
    - cold start makes it harder because we don’t know anything about the user’s interests or if it is a product, we don’t know anything about how well it will do
    - scalability: lots of computations necessary
        - you can write more efficient code or approximate the solution
- models
    - popularity: not personalized, everyone sees the same thing, creates a positive feedback loop
    - classifier: takes in a bunch of input features to predict whether a user will like it or not

Dimensionality Reduction

- data has a lot of dimensions
    - image: a 200×200 image has 12000 features bc rgb
    - text: # of n-grams
    - ratings: one rating for each object
    - course success problem: many features
- why is having too many dimensions a problem?
    - hard to visualize dimensions beyond 3D
    - curse of dimensionality for KNN makes it computationally more intensive
    - overfitting/risk of too much complexity
    - much of the data is redundant/are repeats
- the goal of dimensionality reduction is to reduce the noise in the data while retaining enough signals such that a model is still able to identify meaningful relations between the data
- examples

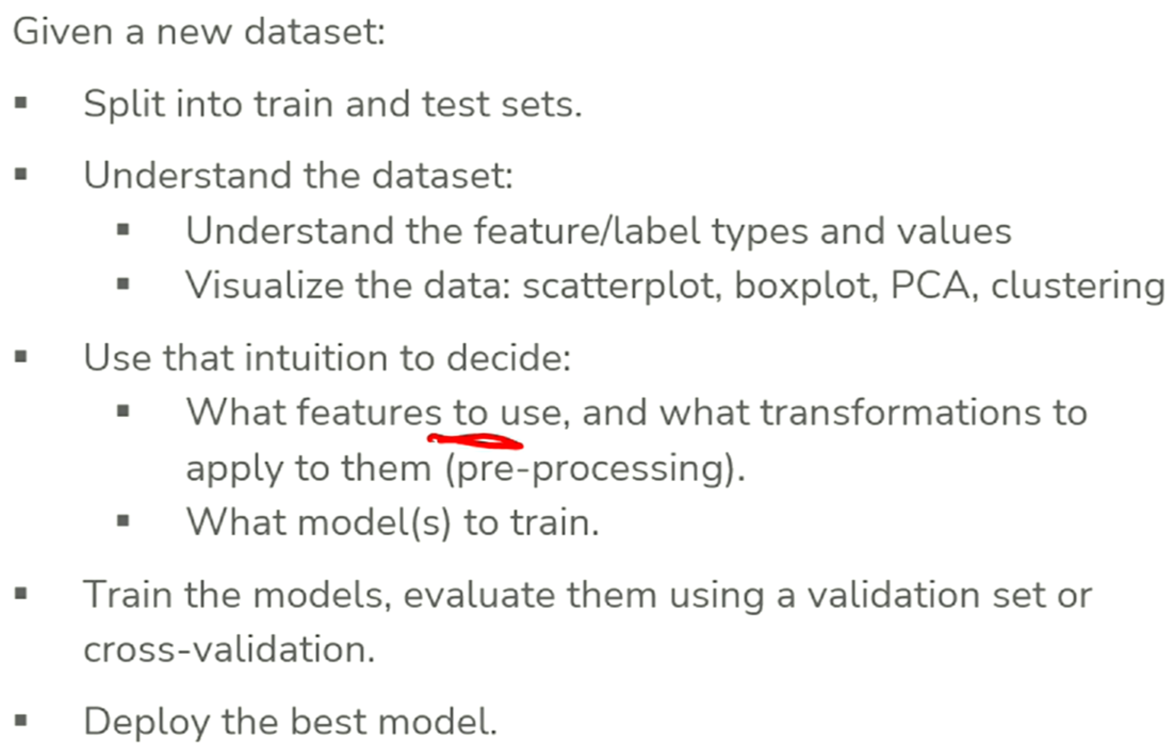
- embedding images in 2d
    - which direction a person is facing + their reaction (stern → happy)
- embedding words in 1D
    - using small number of features still makes it possible to cluster meaningfully

Principal Component Analysis (PCA)

- linearly project a higher dimensional dataset to a lower dimensional one while minimizing the reconstruction error (the error that is created when mapping the lower dimensional dataset back into a higher dimensional dataset)
- linear projection: drop down each point perpendicularly onto the line
    - Z_i=u*x_i
    - 

    - Reconstruction: use only the projection to recreate the original dataset, with some information lost
- 3D data
    - same thing but you plot clusters in 2D
- instead of finding a data point with 3 coordinates, you can now use just 2

- the best project error is the best line of fit that point in the direction with greatest variance
    - you can use 2 eigenvectors to achieve this
    - a higher eigenvalue means that an eigenvector will have a greater variance on that particular axis
- The Algorithm: PCA
    - input data: n*p matrix X
    - center data: subtract the mean from each data point
    - compute spread using the covariance function
    - Select k eigenvectors with the highest eigenvalues
    - project the data onto the principal component using dot products between x and u
    - reconstruct the data and calculate the reconstruction data by subtracting xi^ from xi
- More principal components = higher quality
- example: genes
    - plotting the graphs of the 2 PCAs can sometimes be useful in clustering
- ML Practitioner:
    -


Given a new dataset:

- Split into train and test sets.
- Understand the dataset:
  - Understand the feature/label types and values
  - Visualize the data: scatterplot, boxplot, PCA, clustering
- Use that intuition to decide:
  - What features to use, and what transformations to apply to them (pre-processing).
  - What model(s) to train.
- Train the models, evaluate them using a validation set or cross-validation.
- Deploy the best model.

- Recommender System Setup
  - usually the number of users is greater than the number of products
  - setup a user-item interaction matrix, e.g. for movies you would have user ratings

Matrix Factorization

- user rates a very small amount of movies
- you can create a matrix, where rows are users and columns are ratings for each movi
- most of the matrix will be very sparse
- We have to assume that all movies fit into k categories (such as movie genres)
  - We can create a R_v feature vector with the degree to which each movie fits a particular category
  - Then we can create a user u vector with how much they like each category

    - Then you can put the 2 vectors into a rating function to predict how much a user might like a certain movie
        - dot product of the feature and user vectors
- our goal is to argmin L and R
- some recommender systems
    - popularity
    - nearest neighbor
    - item-item
        - plot a co-occurrence matrix that tells you what percent of people who bought this particular item also bought...
        - you have to normalize first because there might be varying amounts in each category
        - take the top k items in the row that had the highest correlation weights and then recommend that to the user
        - for multiple items:
            - take the average of the scores/ratings of each item for all of those categories
            - could weight recent purchases more than older ones
    - feature-based
        - store information for each feature of each movie AND also the user (age, gender, etc)
    - Matrix factorization
        - actual data points will be the original sparse matrix
        - predict data points by learning the movie and user weights (ratings for each genre, author, date published, time of day, etc) and then dot producting the 2 vectors for the particular user or movie that you want to test
        - suffers from cold start problem

    - You can use ridge regularization to fix the problem of having infinite solutions that minimize the MSE on the quality metric
    - can be used for both supervised and unsupervised learning
  - hybrid model