20211018 Lecture notes

Monday, October 18, 2021 4:39 PM

- Midterm
  - Start as soon as possible
  - Make your first submission as soon as possible, even if it's not the best you can do
  - Unseen test set
  - You are submitting predictions on a test set he has the true value's of
  - Put aside a chunk of your own training set for a validation set
  - Public leader board is only on 30% of the data
    - To avoid overfitting for that 30%
    - To avoid gaming the system
  - Accept Kaggle competition and github classroom
  - Sample.csv
    - Just a sample of what your submission should look like
  - test.csv
    - Your goal is to fill this csv with your best prediction
  - Download all data
  - Clone repo
  - Follow readme
  - Requirements.txt
    - These are not the only packages you can use, you can use whatever you want as long as it's not a deep learning tool
  - Apply model on your test.csv
  - Predict constant
    - Predicts one score for everything
  - Predict KNN
    - a little more formally how you would apply these models
    - With added features you've been extracting
    - You can do some dimensionality reduction, do you want to drop a few columns
    - Plot confusion matrix
  - What can you do during the competition?
    - The bulk of your work should be in feature_extraction.py
    - You should create an exploratory.py file to explore the data
      - Who has watched the most movies
      - Which movies are worst rated
      - What are the top words for reviews of score 5, for

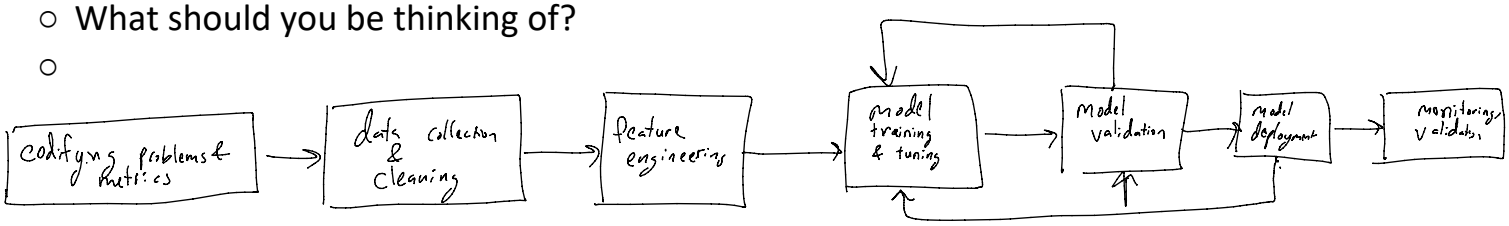
words of score 1?
        - As you start getting familiar with the data, you will get better intuition on what features to extract
        - You have space to be as creative as you want
        - Do you want to look at emojis, capitol letters, punctuation
    - Spend the time to explore the data, extract features based on that exploration
        - Then you can do more with those features
    - SVD may work well, but they can be very expensive
    - TIDF? may be too expensive to run on entire dataset
    - It's OK if your computer can't handle the entire dataset
    - Run given script (test_setup.py) to see if your setup is correct
    - Confusion matrix requires a high version of scipy
    - Right off the bat you can see there is a skew in your data
        - There are more 5.0 than anything else, you may bias your model in a particular way
            - You may want to subset your data, or correct in some other way
    - python3 feature_extraction.py
        - predict…. then make submission to kaggle
        - Then you have the workflow, and all you need to do is make the submission
    - Roll back your code if you make a mistake
    - submission.csv to submit your data
        - Recommend adding comments to your kaggle submission, you will forget which files were from which models
    - 1/2 up the leaderboard just by submitting the default prediction
    - Make a submission as soon as possible
    - Model doesn't need to be complicated, you can do a lot with a simple model with good features
    - Some tools may be deep learning depending on how you use them

==========

Guest speaker:

- Isabel Zimmerman

    - Normally uses google-meet
    - Software engineer for AI center of excellence at RedHat
- What does end to end data science look like?
- Also has QR codes
- Graduating from undergrad spring
- Intern for year at RedHat, rolled over into full time position
- Also a Master's student
- Fully remote work
- AI opps
    - Making data sci easier for people in the cloud
- Data science lifecycle
    - What should you be thinking of?
    -



        - Containers
        - HIVE
        - Normalization
        - Feature engineering
            - R, Python, Juypter hub
            - Garbage in, garbage out
            - More important than model tuning
        - Mostly can get good models using simple methods
            - Deep models often not needed
                - E.g. deep model with 100% accuracy cancer prediction
                    - All of the pictures they were training with had a ruler in the image for cancer samples
                        - Need to make sure you aren't putting in garbage
                        - Avoid biased input
        - Do you need machine learning? It can be expensive
        - Default to the simplest method needed
        - Check to see if you can answer this question without using machine learning
        - Is model performing well?
        - Model deployment
            - Take it out of python
            - How are people interacting your model
            - Model services or Model OPs
            - Are you just updating the training set, or …

- SELDON
  - Creating an API for your model
- Monitoring and validation:
  - Is your application healthy?
  - Look at how data has changed
    - Data drift
      - Global warning
      - Underlying statistical distribution is drifting
  - Prometheus
  - Grafana
- Why should we do data science in the cloud?
  - Juypter hub
  - kafka
  - Grafana
  - Prometheus
  - SELDON
  - All in the container RedHat deploys
  - Scalable
  - Flexible
  - Reproducible
  - All on one platform, so takes care of a lot of the underlying dependecies
- Juypter hub
  - -> Store model -> Google cloud storage/S3/Azure Blob storage/Persistent Volume
    - -> deploy model -> Seldom
      - -> explore model -> juypterhub
      - -> explore metrics -> prometheus
        - -> visualize
- How to try it out?
  - Has some projects data collection and cleaning
  - Link:
    - How to contribute
    - https://www.operate-first.cloud/data-science/ai4ci/
    - https://www.operate-first.cloud/data-science/pet-image-detection/
    - https://www.operate-first.cloud/operations/sre/
- linkedin.com/isabel-zimmerman
- twitter.com/isabel-zimmerman
- (missed some links)

==========

- With large datasets:

    - Sklearn: can train on different batches (pausing…)
    - Simpler models
    - Use cloud/hardware
- Slides will get posted

==========
==========
==========
==========
==========

20211018 Recommender Systems 20211020 add on

Monday, October 18, 2021 4:28 PM

- Related to midterm, will get into it more Weds

An Example: Movie Recommendations

Given

- Users: $U_1, \ldots, U_n$
- Movies: $M_1, \ldots, M_m$
- Ratings: R_{ij}

Goal: Recommend movies to users

Challenges:

- Scale (millions of users, millions of movies)
- Cold Start (change in user base, change in content)
- Sparse Data (Not many users rank movies)

- Cold start:
  - If you want to start a product, you need users in order to recommend movies to your users
  - How do you get started?
- Sparse data
  - Not much information to begin with
  - Could use proxies
    - Did you binge over a weekend, or only watch the first 5 minutes?

An Example: Movie Recommendations

	M_1	M_2	M_3	M_4
U_1	R_{11}	R_{12}	R_{13}	R_{14}
U_2	R_{21}	R_{22}	R_{23}	R_{24}
U_3	R_{31}	R_{32}	R_{33}	R_{34}

Use Rating prediction as proxy for recommendation!

- Going to try to predict these ratings, as this is a proxy for recommendations

An Example: Movie Recommendations

	M_1	M_2	M_3	M_4
U_1	5	?	0	0
U_2	?	4	0	0
U_3	0	?	4	?

- Want to predict the score a user would give to a particular movie

An Example: Movie Recommendations

	M_1	M_2	M_3	M_4
U_1	5	5	0	0
U_2	5	4	0	0
U_3	0	0	4	5

Neighborhood Methods

- (user, user) similarity measure
  - i.e. recommend same movies to similar users (requires info about users)
- (item, item) similarity measure
  - i.e. recommend movies that are similar (requires info about movies)

Pros:
- Intuitive / easy to explain
- No training
- Handles new users/items

Challenges:
- Users rate differently (bias)
- Ratings change over time (bias)

- 3 ways to approach this problem
- Recommend movies that are similar to one another
- Strength of recommendation will depend on these measures
- Hard to come up with this before you have any user base
- Can't know ahead of time exactly what features you need
  - You could have a good guess for some of them, like genre
- Data drift over time could cause users to rate movies different over time

- This is a totally viable solution if …
- First easiest thing we can do
- We recommend this, because you enjoyed … or users similar to you enjoyed …
- Ratings changing over time
    - Users may be similar today but in a few years they might not be
- This should be something you can implement for the midterm
    - KNN is in the starter code they gave us
- Annotate features you need
- Try to relax assumptions by doing content based filtering
- Will also try to learn the features automatically

Feature Extraction - Content-Based

Realistically:

- It's difficult to characterize movies and users with the right features
- Characterization of users and movies may not be accurate
  - If you are using genres for example, movies with varying degree of "comedy" will get the tag "comedy".

Goal:

- Discover the best features in an automated way

Content-Based: assume you have features for movies - want to learn features for users

Collaborative filtering: want to learn features for both users and movies

---

- Relaxing the features you need
- Don't ask users to fulfill a giant questionnaire
  - Content-based
- Collaborative:
  - Use an algorithm
- There may be nuances to how comedic a movie is
  - Need to be flexible, and it's hard to do this manually
  - Can do this in an automated way

Feature Extraction - Content-Based

Suppose we have a set of features that characterizes each movie (ex: category, genre...), we could obtain the following **feature-to-movie** similarity matrix:

	M_1	M_2	M_3	M_4
F_1 (Romance)	.9	1	.1	0
F_2 (Action)	0	.01	1	.9

-

Feature Extraction - Content-Based

Given this **feature-to-movie** similarity matrix, how can we predict rating for User 2 or Movie 1 (i.e. R_{12})?

If we had a **user-to-feature** similarity matrix, we could multiply:

user-to-feature x feature-to-movie = user-to-movie = R_{ij}

user-to-feature x feature-to-movie = user-to-movie = R_{ij}

- What we would like to do is
- Reminds you of decomposition of matrix into two matrices
  - This should you remind you of singular value decomposition
- We can decompose the matrix with missing values
  - So we will formulate our decomposition in a different way

Feature Extraction - Content-Based

	F_1 (Romance)	F_2 (Action)
U_1	5	0
U_2	5	0
U_3	0	5

X

	M_1	M_2	M_3	M_4
F_1 (Romance)	.9	1	.1	0
F_2 (Action)	0	.01	1	.9

Feature Extraction - Content-Based

$$P^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$Q^{(1)} = \begin{bmatrix} .9 \\ 0 \end{bmatrix}$$

$$R_{21} = P^{(2)T} \cdot Q^{(1)}$$

$$= \begin{bmatrix} 5 & 0 \end{bmatrix} \cdot \begin{bmatrix} .9 \\ 0 \end{bmatrix}$$

$$= 4.5$$

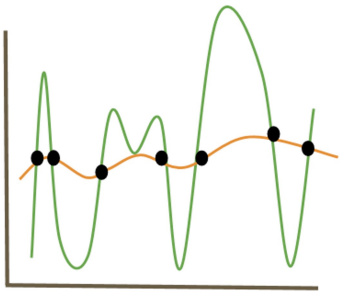
But, how to we find $p^{(1)}$, ... , $p^{(n)}$?

- We want to formulate this in a way to algorithmically find the p's

Feature Extraction - Content-Based

$$P^{(j)} = \arg\min_{P} \frac{1}{\|M^{(j)}\|} \sum_{i \in M^{(j)}} (P^T Q^{(i)} - r_{ij})^2 + \lambda \|p\|^2$$

Regularization Term: a penalty on the size of the parameter p



- We want to find a Matrix P that minimizes the score
- Regularization
  - Penalty term to penalize over complexity
  - We want to see the orange curves, not the green curves
    - The green curve would need a lot of parameters to be able to predict that complexity
  - lambda is a regularization parameter that we'll have to tune
  - P x Q should be close to the scores we've seen so far

- This is relatively complicated to implement
  - So we won't cover how to solve this
- We want a good approximation for the scores we have

Feature Extraction - Collaborative Filtering

Challenge with content-based:

How to get the right features $f_1, ..., f_k$ **and** $p^{(1)}, ... , p^{(n)}$?

Can we learn these features?

$$R = PQ$$

- Now we want to find P & Q at the same time
  - Previously we had P
  - Now we don't have P and Q
  - Now we have a sparse matrix

Feature Extraction - Collaborative Filtering

Can't use SVD because R is sparse... BUT, we can formulate an optimization problem to solve:

$$\min_{p,q} \sum_{i,j \in R} (r_{ij} - p_i^T q_j)^2 + \lambda(\|p\|_F^2 \|p\|_F^2)$$

Should be q

To solve, take derivatives wrt P & Q. Then, just like Expectation-Maximization Algorithm from GMM:

1. Start with random Q
2. Get P
3. Improve Q
4. Repeat 2 & 3

- We can extend our optimization from before
- This is not dissimilar to maximization optimization algorithm
- Based on improvement, you can keep re-estimate
- Take partial derivatives with respect to P and Q
  - Take random, then estimate other and back and forth
- If you can implement this for the midterm, great
  - BUT our matrix is likely too big
- You have unknown P and Q when you don't want to come up with the features for Movies nor Users
  - This is the most hands off, doesn't require

users to fill out form, but ALSO doesn't require the admins also to come up with features

20211018 SVM-and-boosting

Monday, October 18, 2021 5:29 PM

- If you have 25 classifiers, and they are all independent, what is the probability of making a mistake?
  - Error rate of 35% -> 6%
- Bagging
  - Sampling with replacement
- Look at SVM code
  - Look at bagging/boosting classifier
  - Pool that
  - Code already posted on github, check it out
- Boosting
  - Expose model to examples it is not good let
    - Start with equal weights
    - Then bump probability of being selected from those examples
  - Round 1: sample randomly
  - Round 2: Add in more 4s
  - Round 3: Add even more 4s
  - You can end up with a situation where you accidentally added only the item in the model is the thing it can't learn
- AdaBoost
  - Favor classifiers that are doing better rather than classifiers that are not
  - Try to use these ensemble methods
  - There are packages in Sklearn to do this