Collaborative Filtering

Piyawat L Kumjorn & Panida Nimnual

Features to prediction

- User's Past Interactions (Tweet / Retweet)
  - User Modeling
- Tweet Messages
  - Content-based algorithm
- Relationship between Users
  - Collaborative Filtering
- Influential User
  - Influential Ranking Analysis

Agenda

- Information Filtering
- Collaborative Filtering
  - Memory-based
  - Model-based
- Evaluation Metrics
- Twitter Model

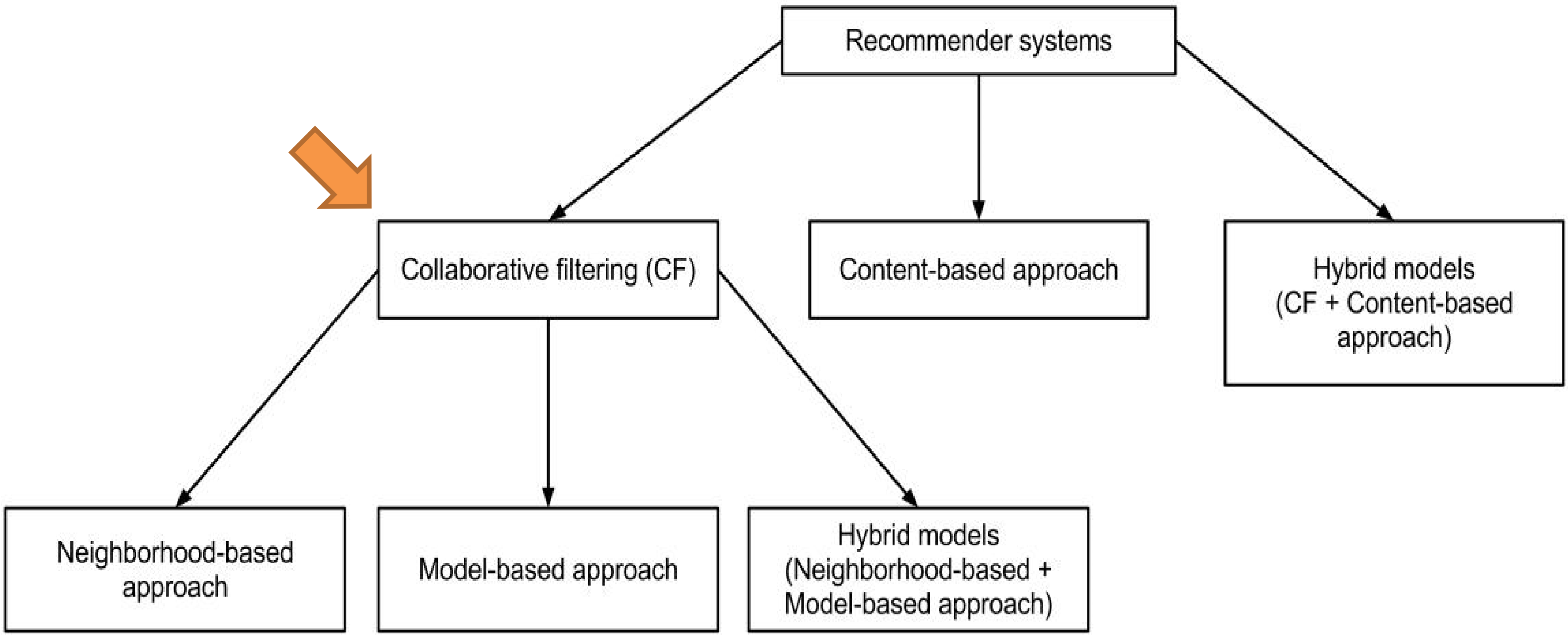
Information Filtering

- Problem: Delivery of information that the user is likely to find interesting or useful
- It can be called a recommender system
- The system must be personalized
- This requires the gathering of feedback from the user to make a profile of the his preferences

Information Filtering

- Two major approaches for information filtering
  1. Content-based filtering : content of the items and the user's preferences
  2. Collaborative filtering : the correlation between people with similar preferences
- Hybrid systems = Content-based + Collaborative
- Alternative approaches
  - Demographic Filtering
  - Economical Filtering

Overview of Collaborative Filtering



Memory-based approach

http://en.wikipedia.org/wiki/Collaborative_filtering

Memory-based Approach

http://en.wikipedia.org/wiki/Collaborative_filtering

Memory-based Approach

- E.g. Movie Ratings

User-based Filtering

	Amy	Jeff	Mike	Chris	Ken
The Piano	-	-	+		+
Pulp Fiction	-	+	+	-	+
Clueless	+		-	+	-
Cliffhanger	-	-	+	-	+
Fargo	-	+	+	-	+

Item-based Filtering

http://recommender-systems.org/collaborative-filtering/

Memory-based Approach

– A prediction is normally based on the weighted average of the recommendations of several people.

Find Similarity → Weighted Prediction

Similarity Computation

– Correlation-Based Similarity

- *Pearson correlation*

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}}$$

u, v are users *i is an item*

I is the set of all items that b

$r_{u,I}$ is a rating that user u give

$\bar{r}_u$ is an average rating given

Other correlation similarity:

- Constrained Pearson correlation
- Spearman rank correlation
- Kendall's τ correlation

Similarity Computation

– Vector Cosine-Based Similarity

$$w_{u,v} = \cos(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

u, v are users

$\vec{u},\vec{v}$ *are vectors of rating scores that u and v have rated respectively*

Similarity Computation

– Example

	Chris	Ken
The Piano		+1
Pulp Fiction	-1	+1
Clueless	+1	-1
Cliffhanger	-1	+1
Fargo	-1	
Mean	-1/3	+1/3

– Correlation

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}}$$

$$w_{u,v} = \frac{(-1+1/3)(1-1/3)+(1+1/3)(-1-1/3)+(-1+1/3)(1-1/3)}{\sqrt{2(-1+1/3)^2+(1+1/3)^2}\sqrt{2(1-1/3)^2+(-1-1/3)^2}}$$

$$w_{u,v} = -1$$

– Cosine-based

$$w_{u,v} = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

$$w_{u,v} = \frac{(-1)+(-1)+(-1)}{\sqrt{1+1+1}\sqrt{1+1+1}} = \frac{-3}{3} = -1$$

Weighted Prediction

- Simple Weighted Average

$$P(r_{a,i}) = \frac{\sum_{u \in U} w_{a,u} r_{u,i}}{\sum_{u \in U} | w_{a,u} |}$$

- Weighted Sum of Others' Ratings

$$P(r_{a,i}) = \bar{r}_a + \frac{\sum_{u \in U} w_{a,u} (r_{u,i} - \bar{r}_u)}{\sum_{u \in U} | w_{a,u} |}$$

$P(r_{a,i})$ is the prediction value of rating that user a give to item i

$w_{a,u}$ is the similarity weight of user a and user u

$\bar{r}_a$ Is the average rating score of user a giving to all items

Memory-based Approach

- Many collaborative filtering systems have to handle a large number of users. So, selecting some nearest neighbors for computation can improve the performance.

Find Similarity → Select neighbors → Weighted Prediction

- Two techniques
    - Correlation-thresholding (who's correlation is greater than a given threshold)
    - Best-n-neighbors (with the highest correlation)

Sparsity Problem

- Most collaborative filtering systems have to deal with too few ratings.
- Occurs when number of users and items are very large
- Two people have few rated items in common making the correlation coefficient less reliable
- Several solutions have been proposed:
  - *Implicit ratings (Missing Value Imputation)*
  - *Dimensionality reduction*
  - *Content description*

Classification Approach

– Collaborative filtering can also be formulated as a classification problem

	Amy	Jeff	Mike	Chris	Ken
The Piano	-	-	+		+
Pulp Fiction	-	+	+	-	+
Clueless	+		-	+	-
Cliffhanger	-	-	+	-	+
Fargo	-	+	+	-	?

http://recommender-systems.org/collaborative-filtering/

Classification Approach

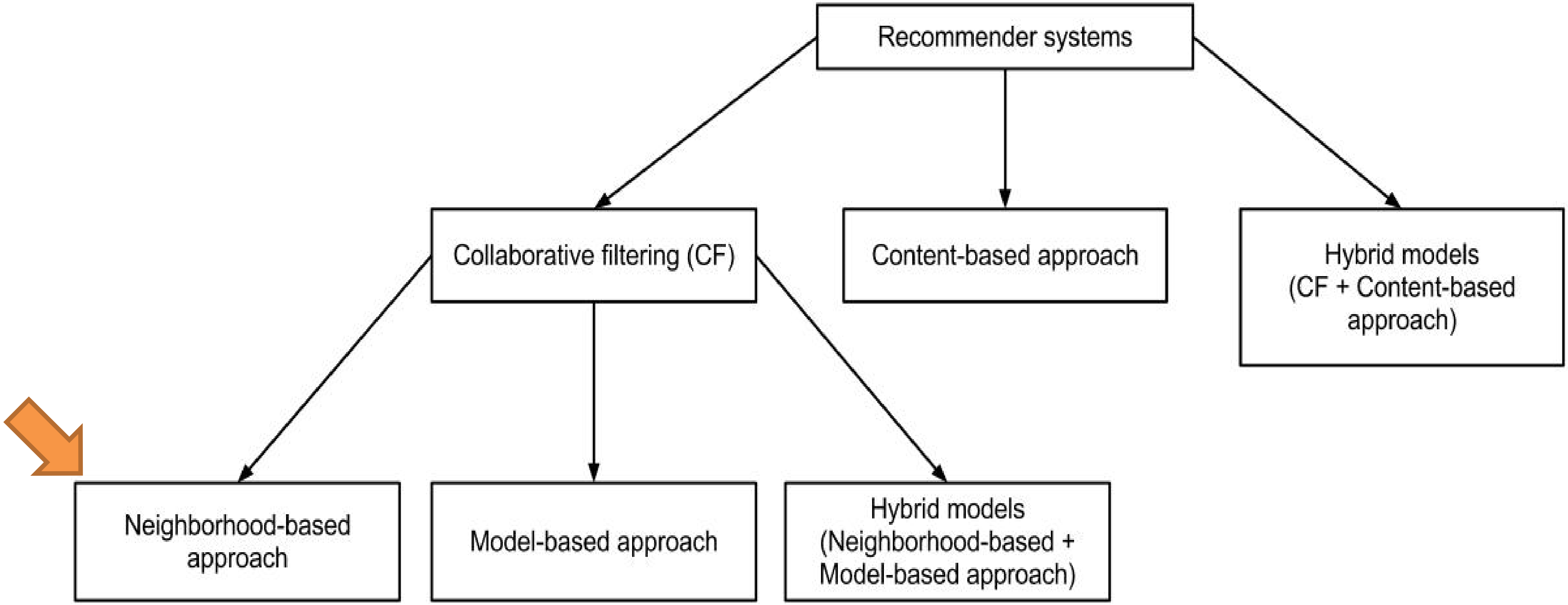
	The Piano	Pulp Fiction	Clueless	Cliffhanger	Fargo
Amy +	0	0	1	0	0
Amy -	1	1	0	1	1
Jef +	0	1	0	0	1
Jef -	1	0	0	1	0
Mike +	1	1	0	1	1
Mike -	0	0	1	0	0
Chris +	0	0	1	0	0
Chris -	0	1	0	1	1
Class	+	+	-	+	?

http://recommender-systems.org/collaborative-filtering/

Extensions to Memory-based

- *Default Voting*
- *Inverse User Frequency*
- *Case Amplification*
- *Imputation-Boosted CF Algorithms*
- *Weighted Majority Prediction*

Overview of Collaborative Filtering



Memory-based
approach

http://en.wikipedia.org/wiki/Collaborative_filtering

Model-based Approach

- Models are developed using data mining, machine learning algorithms to find patterns based on training data
- Examples
  - Bayesian networks
  - Clustering models
  - Latent semantic models
  - Regression-based models
  - Markov Decision Processes
  - Etc.

Clustering Models

– Group users into classes. Users who are in the same class have same interests.

– Apply obtained clusters in many ways :

- use a memory-based CF algorithm to make predictions within each cluster
- The *RecTree*, using k-means with k = 2, recursively splits the large rating data into two sub-clusters as it constructs from the root to its leaves. Each internal node maintains rating centroids of their subtrees. The prediction is made within one specific leaf node.
- Using naïve bayes principle to find rating scores

Latent Semantic Models

– A *Latent semantic CF* introduces latent class variables in a mixture model allowing us to discover the latent features underlying the interactions between users and items

Latent Semantic Models

– E.g. Movie Ratings

	i_1	i_2	i_3	i_4	i_5	…	i_n
u_1	2.7						3.2
u_2				5.[illegible]			
u_3							
[illegible]							
u[illegible]			[illegible]				
…							
u_m				1.4			1.5

There should be some latent features that determine how a user rates an item

m users

n movies

m x n matrix

R : m x n matrix

Latent Semantic Models

K features

Features: Erotic love scenes, Black Comedy, Famous of actors, Time Period

	f_1	f_2	f_3	…	f_k
u_1	0.8	-0.3	-3.3		2.4
u_2	-3.9	-2.6	-0.1		-3.4
u_3	4.2	3.6	-1.0		3.3
u_4	0.2	3.4	1.1		-2.4
u_5	-3.2	-3.8	3.7		3.7
…					
u_m	-3.4	2.6	4.0		1.0

P : m x k matrix

Features: Erotic love scenes, Black Comedy, Famous of actors, Time Period

	f_1	f_2	f_3	…	f_k
i_1	3.7	-2.2	-1.9		0.6
i_2	-1.1	1.0	-0.5		-3.1
i_3	-3.8	2.8	3.0		3.9
i_4	0.2	4.4	3.2		-3.9
i_5	-0.4	-4.9	-1.6		-4.2
…					
i_n	1.9	0.8	-2.5		2.6

Q : n x k matrix

Latent Semantic Models

– To find R : m x n matrix

$P_{m \times k}$

	f_1	f_2	f_3	…	f_k
u_1	0.8	-0.3	-3.3		2.4
u_2	-3.9	-2.6	-0.1		-3.4
u_3	4.2	3.6	-1.0		3.3
u_4	0.2	3.4	1.1		-2.4
u_5	-3.2	-3.8	3.7		3.7
…					
u_m	-3.4	2.6	4.0		1.0

X

$Q^T_{k \times n}$

	i1	i2	i3	i4	i5	…	in
f1	3.7	-1.1	-3.8	0.2	-0.4		1.9
f2	-2.2	1	2.8	4.4	-4.9		0.8
f3	-1.9	-0.5	3	3.2	-1.6		-2.5
…							
fk	0.6	-3.1	3.9	-3.9	-4.2		2.6

$$R \approx P \times Q^T = \hat{R}$$

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

Matrix Factorization

- Single Value Decomposition (SVD)
- Stochastic Gradient Descent
- Alternating Least Square

Stochastic Gradient Descent

- For each user-item pair

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

- The gradient at the current values

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2 e_{ij} q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2 e_{ij} p_{ik}$$

- Gradient Descent for each iteration

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

Stochastic Gradient Descent

- For learning process, we train only user-item instances (u_i , i_j , r_{ij}) rated in the training dataset.
- To stop iterations, we may consider the overall error.

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2$$

- Introducing regularization to prevent overfitting

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2 + \frac{\beta}{2}\sum_{k=1}^{K}(||P||^2 + ||Q||^2)$$

- The new update rules are as follows.

$$p'_{ik} = p_{ik} + \alpha\frac{\partial}{\partial p_{ik}}e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$$
$$q'_{kj} = q_{kj} + \alpha\frac{\partial}{\partial q_{kj}}e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$$

http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

Implementation in Python

```
import numpy

def matrix_factorization(R, P, Q, K, steps=5000, alpha=0.0002, beta=0.02):
    Q = Q.T
    for step in xrange(steps):
        for i in xrange(len(R)):
            for j in xrange(len(R[i])):
                if R[i][j] > 0:
                    eij = R[i][j] - numpy.dot(P[i,:],Q[:,j])
                    for k in xrange(K):
                        P[i][k] = P[i][k] + alpha * (2 * eij * Q[k][j] - beta * P[i][k])
                        Q[k][j] = Q[k][j] + alpha * (2 * eij * P[i][k] - beta * Q[k][j])
        eR = numpy.dot(P,Q)
        e = 0
        for i in xrange(len(R)):
            for j in xrange(len(R[i])):
                if R[i][j] > 0:
                    e = e + pow(R[i][j] - numpy.dot(P[i,:],Q[:,j]), 2)
                    for k in xrange(K):
                        e = e + (beta/2) * (pow(P[i][k],2) + pow(Q[k][j],2))
        if e < 0.001:
            break
    return P, Q.T
```


http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

Comparison

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD, PCA*	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

Evaluation Metrics

- Predictive accuracy metrics
  - *Mean Absolute Error (MAE) and its variations*
- Classification accuracy metrics
  - *Precision, recall, F1-measure, and ROC sensitivity*
- Rank accuracy metrics
  - *Pearson's product-moment correlation*
  - *Kendall's Tau*
  - *Mean Average Precision (MAP)*
  - *Half-life utility*
  - *Normalized distance-based performance metric (NDPM)*

Predictive accuracy metrics

The lower, the better

– Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{\{i,j\}} \left| p_{i,j} - r_{i,j} \right|}{n}$$

– Normalized Mean Absolute Error (NMAE)

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}}$$

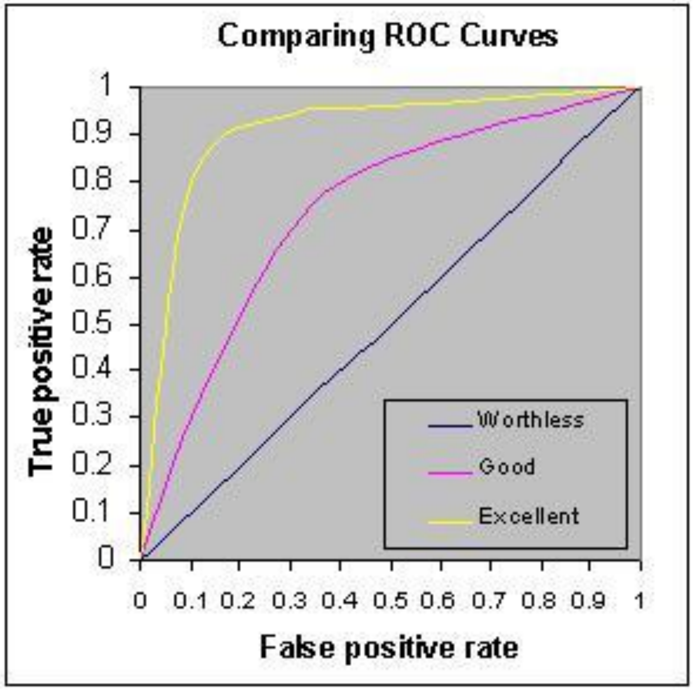
– Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n} \sum_{\{i,j\}} \left(p_{i,j} - r_{i,j} \right)^2}$$

Classification Accuracy Metrics

– ROC Curve
  - X-axis : True Positive Rate (Recall) = TP / (TP+FN)
  - Y-axis : False Positive Rate = FP / (FP+TN)
  - Area Under Curve (AUC)
  - A bigger *AUC value is better.*

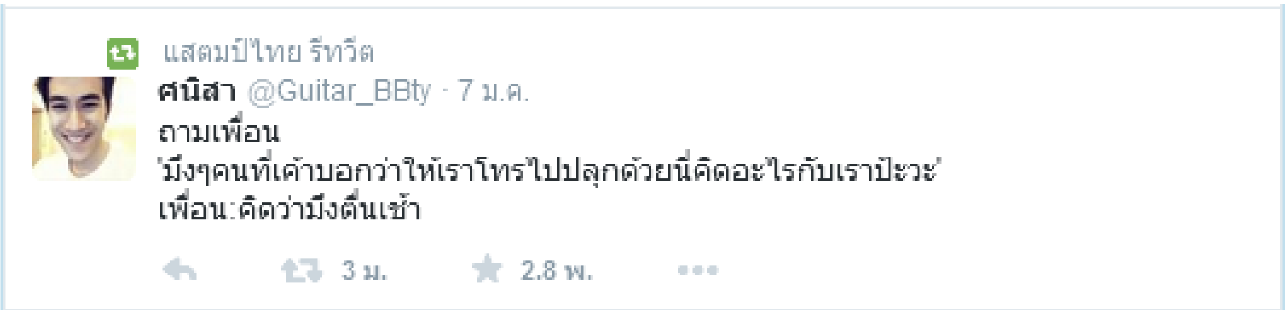
Actual	Predicted	
	Positive	Negative
Positive	TruePositive	FalseNegative
Negative	FalsePositive	TrueNegative



http://gim.unmc.edu/dxtests/roc3.htm

Twitter Model

- Rating in twitter may refer to retweet, reply, and favorite interaction. Each of these kinds is considered as a binary value.



- Retweet and Reply generate new tweet.
- Each user cannot see all tweets.

Associated Twitter API

- GET statuses/retweets/:id
  - Returns a collection of the 100 most recent retweets of the tweet specified by the id parameter.
- GET statuses/home_timeline
  - Returns up to 800 Tweets and retweets posted by the authenticating user and the users they follow.
- GET followers/ids
  - Returns up to 5,000 user IDs for every user following the specified user.

https://dev.twitter.com/rest/public

Associated Twitter API

- GET friends/ids
  - Returns up to 5,000 user IDs for every user the specified user is following.
- GET statuses/user_timeline
  - Returns up to 3,200 of most recent Tweets posted by the specified user.
- GET favorites/list
  - Returns the 200 most recent Tweets favorited by the authenticating or specified user.

https://dev.twitter.com/rest/public

Collecting Data

Random k users
(k = 20)

→ Expand the community by their friends and followers

→ Get all posts and favorites of the members from the community

→ Select n users that are interested in tweets of more than 15 members

→ Predict interaction of these n users