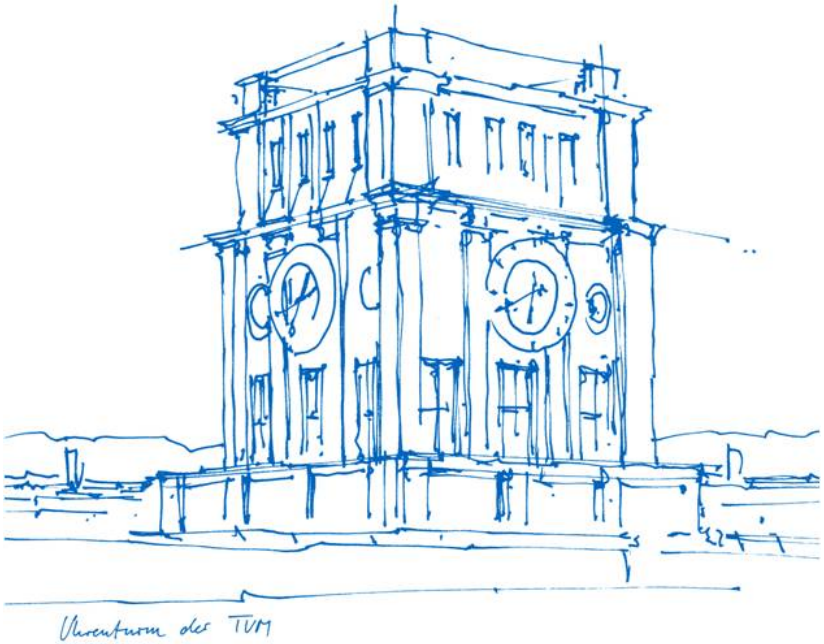
Exercises for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to Exercise 6



Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python: basic Python programming language exercises • Graph Drawing using igraph	Monday, May 27 - Monday, June 3, 24:00
2	• Centrality measures	Monday, June 3 - Monday, June 17, 24:00
3	• Recommender Systems as an example for systems using simple forms of social context: Collaborative Filtering	Monday, June 17 - Monday, June 24, 24:00
4	• Inferring social tie strength from activity data in social networking platforms with linear regression	Monday, June 24- Monday, July 01, 24:00
5	• finding social groups with clustering methods on profile date (K-Means) and graph clustering (Girvan Newman method)	Monday, July 01 - Monday, July 08, 24:00
6	• analyzing short-term social context using mobile interaction data (Reality Mining)	Monday, July 08 - Monday, July 15, 24:00

Reality Mining

- Alex Pentland at MIT Media Lab: Reality Mining:
  - collect human behavior data via sensors
  - long term, medium term, short term behavior data
  - individual context as well as social context
  - built models from the data via machine learning
  - → computational social science as well as social computing

- Reality Mining dataset (2005):
  - 9 months, 106 participants (MIT staff and students)
  - mobile phone data: call logs, device's location status, other Bluetooth devices in proximity, devices within the same cell tower range
  - surveys among participants

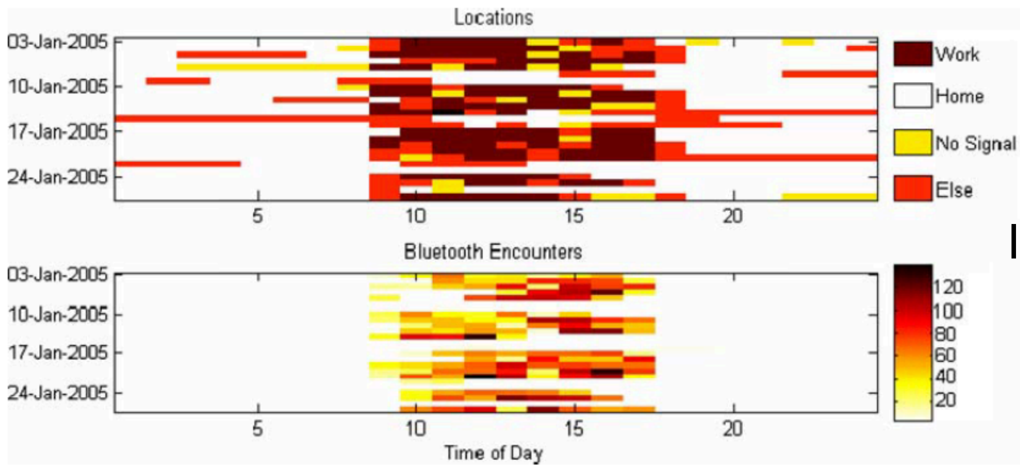
Problem 6.1: Activity Patterns & Entropy

- data: coarse average locations of users per hour
  - per user + per hour of day (0-23):
    - 0 – no signal
    - 1 – at home
    - 2 – at work
    - 3 – elsewhere
    - NaN – phone is off

- possible application: location prediction (e.g. using RNNs or HMMs)

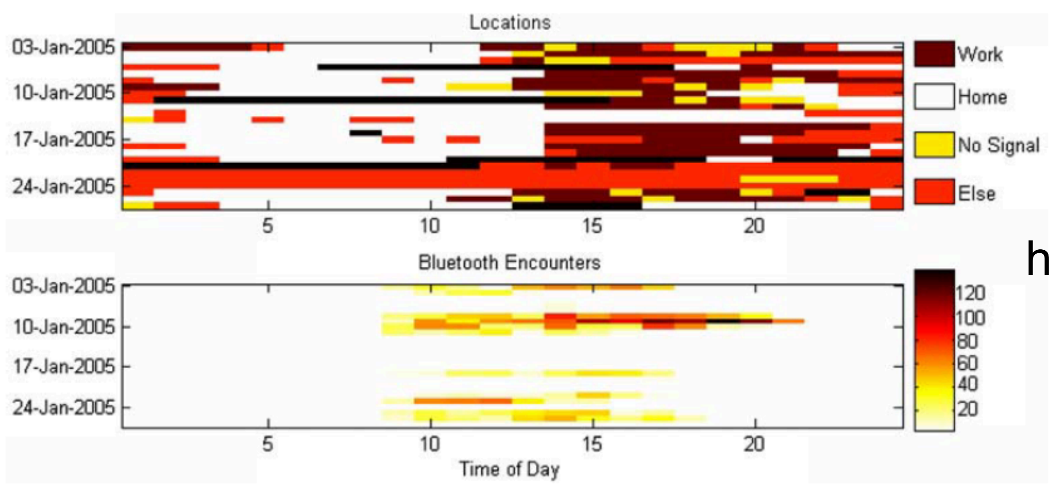
Problem 6.1: Activity Patterns & Entropy

- measure for predictability: entropy: $H(x) = -\sum_{n=1}^{N} p(x) \log p(x)$
  - e.g. for the 4 location cases above



low entropy person

[1]

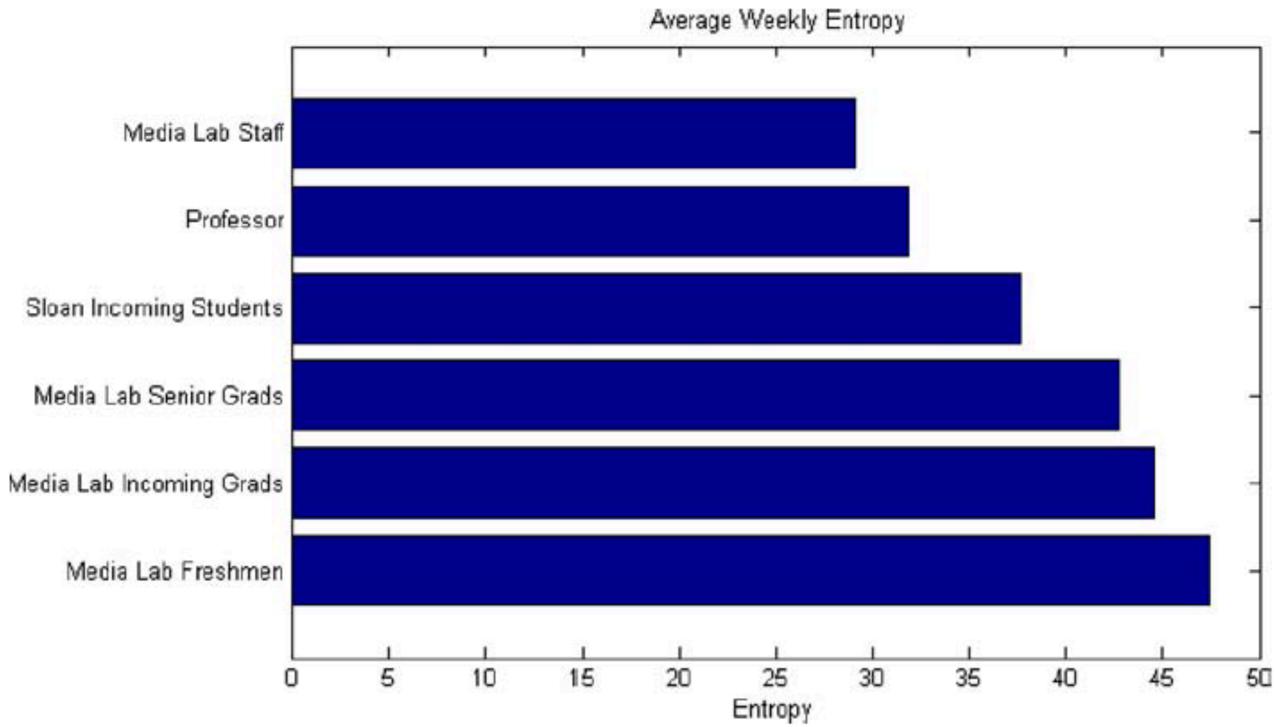


high entropy person

[1]

Problem 6.1: Activity Patterns & Entropy

- measure for predictability: entropy: $H(x) = -\sum_{n=1}^{N} p(x) \log p(x)$
  - e.g. for the 3 location cases above



[1]

Problem 6.1: Activity Patterns & Entropy

Task 1: Entropy Calculation

Compute the entropy values for all users for the whole duration of the study. For this purpose, count the occurences of each location for every user and calculate a probabilty distribution of the places. Then implement the formula given above with the just calculated probabilities. As mentioned above, not all UUIDs from 0 to 106 are taken so you should work with an exception here in order to avoid key errors - assign an entropy of -1 if an UUID is missing. Your program's output is supposed to be an entropy list, containing the values for each user.

Task 2: Predicability of Activity Patterns

The next step is to compare the daily activities of two different subjects, one with a low entropy and one with a much higher entropy, and observe the differences in the regularity of their patterns. **Choose one subject with a low and another with a high entropy value, get their hourly locations for one month and create a heatmap for each. Describe the subjects' daily activites and the disparities between both subjects. Explain the resulting relationship between entropy and the predictability of an individual's activity patterns.** Do not write more than 5 sentences.

Notes:

- Due to the lack of data for some users, their data might not be as insightful on certain months as for other subjects - choose accordingly.
- It is helpful to fill NaN values with -1.0 using pandas `fillna` function.

Task 3: Prediction Applications

After having discovered the predictability in the first place, one could compute an above mentioned model (e.g. with the help of an HMM as done in [1]) in order to infer the locations of one or more users. **Which applications do these prediction have, what could they be used for? Discuss their advantages and disadvantages.** Don't write more than 5 sentences.

TODO: Your discussion here!

Inferring Long-Term Social Context from Short-Term Social Context

- goal: predict friendship (yes, no) from short-term social context features:
  - totalprox - total encounters with other subjects (min/day)
  - satprox – encounters with other subjects on a Saturday night (min/week)
  - nosignalprox – time where other subjects did also not have a signal (min/day)
  - homeprox – encounters with other subjects at home (min/week)
  - commoncell – total number of common cell towers for two subjects
  - callevent – number of phone calls per day from others for each user

- ground truth: from survey:

  isfriend – 1 if the subject considers the other their friend, 0 if not

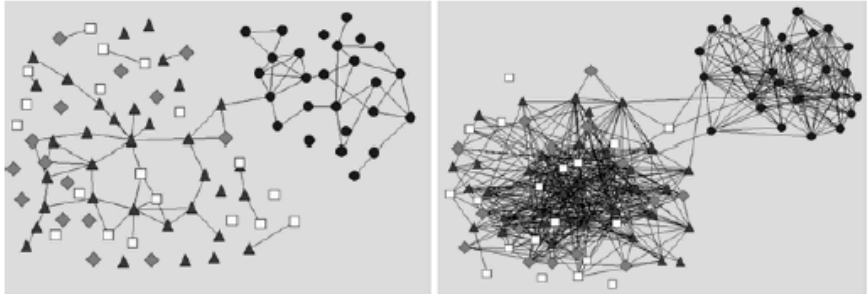


Fig. 11 Friendship (*left*) and daily proximity (*right*) networks. *Circles* represent incoming Sloan business school students. *Triangles*, *diamonds* and *squares* represent senior students, incoming students, and faculty/staff/freshman at the Media Lab. While the two networks share similar structure, inferring friendship from proximity requires the additional information about the context (location and time) of the proximity

[1]

Problem 6.2: Inferring Friendship with GMM Clustering

- attempt to cluster the short-term feature vectors for each person into two clusters with a two Gaussian GMM $(K = 2)$
- use the Gaussian with the lower a priori probability (π) as the „non-friend“ Gaussian.
- compute the accuracy for the friendship prediction resulting from that approach by comparing it with the ground truth

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

x:

- totalprox - total encounters with other subjects (min/day)
- satprox – encounters with other subjects on a Saturday night (min/week)
- nosignalprox – time where other subjects did also not have a signal (min/day)
- homeprox – encounters with other subjects at home (min/week)
- commoncell – total number of common cell towers for two subjects
- callevent – number of phone calls per day from others for each user

Problem 6.2: Inferring Friendship with GMM Clustering

(1) Since we have the actual friendship network available, the results can be compared with that network graphically. We are also interested in the accuracy, the proportion of correctly predicted friends. **First of all, complete the code by computing the accuracy. What does the outcome indicate? Compare the actual and the inferred friendship graph.** Don't write more than 3 sentences.

(2) As a next step, you are supposed to deal with the GaussianMixture module. Change the number of compontents in the `GaussianMixture` class constructor to other values. **What reasoning does a GMM use to predict labels? How does the inferred graph change? Can you think of any implications for the distinction of friendships and are there other features suitable to get meaningful information?** Don't write more than 5 sentences.

Note: The problematic with predicting labels from a GMM is not knowing what label belongs to which component - we assume the most commonly predicted label is always 'not friends' which is more realistic.

Problem 6.3: Inferring Friendship with Generative Classification

- clustering based approach is not so good → build true classifier
- two classes ($y \in$ {friendship, no friendship})
- input x as before

Discriminative Classifiers

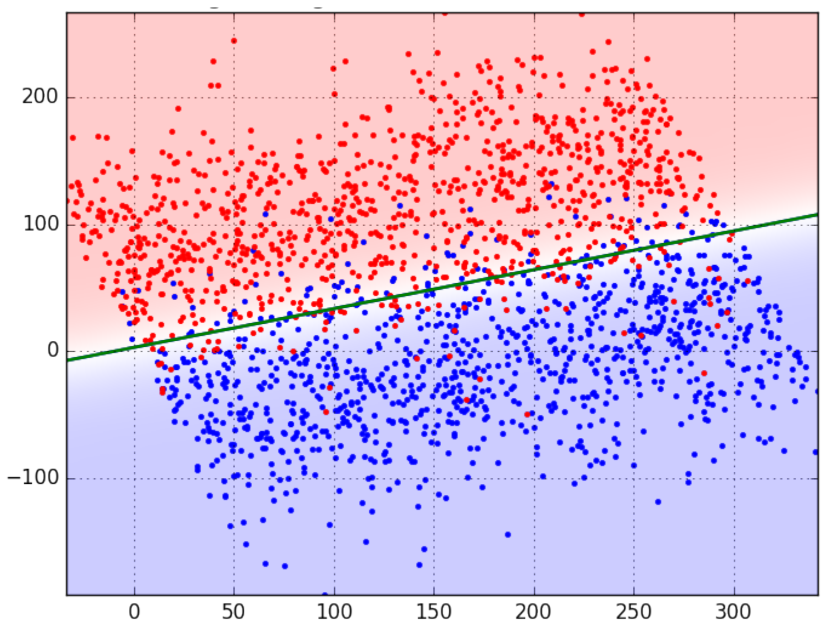
- two class discriminative classifiers: learn discrimination surface between two classes directly, i.e. model is $p(y|x,\theta)$

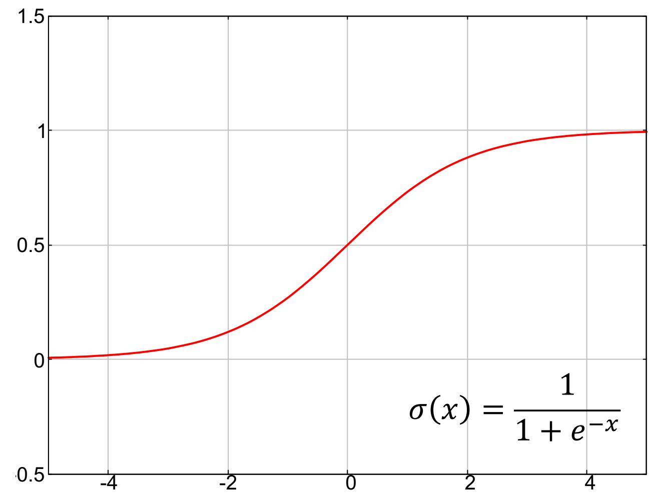
- → optimize $\mathrm{L}(\theta) = \prod_{n=1}^{N} p(y^{(n)}|x^{(n)},\theta)$ or more usually $\log \mathrm{L}(\theta) = \sum_{n=1}^{N} \log p(y^{(n)}|x^{(n)},\theta)$ directly w.r.t. θ .

example: logistic regression:

$$p(y=1|x,\theta=w) = \sigma(w^T x)$$

$$p(y=0|x,\theta) = 1-\sigma(w^T x)$$





[2]

Generative classifiers

- two class generative classifiers:
  $p(y|x,\theta) \propto p(x|y,\theta)\, p(y|\theta)$
  → learn the class conditional density $p(x|y,\theta)$ and the class priors $p(y|\theta)$

- → optimize $\mathrm{L}(\theta) = \prod_{n=1}^{N} p(x^{(n)}|y^{(n)},\theta)\, p(y^{(n)}|\theta)$ or more usually $\log \mathrm{L}(\theta)$ w.r.t. θ .

- we will use a GMM for each of the two class conditional densities
  $p(x|y=i,\theta) = \sum_{k=1}^{K(y=i)} \pi_k^{(y=i)} \mathcal{N}(x|\mu_k^{(y=i)}, \Sigma_k^{(y=i)})$ (for $i \in \{0,1\}$)

- and a Bernoulli distribution for the class priors
  $p(y|\theta) = \Theta^y(1-\Theta^{1-y})$

Classifiers: Error / Performance Measures: Two Classes

Accuracy:

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

		Actual	
		y=1	y=0
Predicted	y=1	TP	FP type I error
	y=0	FN type II error	TN

Precision (positive predictive value):

$$PREC = \frac{TP}{TP + FP}$$

Recall (sensitivity, true positive rate):

$$REC = \frac{TP}{TP + FN}$$

Specifity (true negative rate):

$$TNR = \frac{TN}{FP + TN}$$

False Negative Rate (miss rate):

$$FNR = \frac{FN}{TP + FN}$$

False Positive Rate (fall out):

$$FPR = \frac{FP}{FP + TN}$$

F1 Score (harmonic mean of Recall and Precision):

$$\text{F1} = \frac{2 * PREC * REC}{PREC + REC}$$

Problem 6.3: Inferring Friendship with Generative Classification

The Task

For this task, we use the previously read in variables `friends_set` and `not_friends_set` which corresponds to the data in `feature_table` but as a preprocessed format for easier usage in the below classification. The first set contains the people that consider another person their friend and the respective feature values that were already used in Problem 6.2, accordingly for the second set.

Complete the code in the `GMM_classify` according to the formula given above. Vary the number of components as well as the number of feature vectors for training and testing, and evaluate the resulting classifer using the statistical measures presented above. What is the intuitive meaning for them in our case? Don't write more than 6 sentences.

Notes:

- It might be helpful to look at the class priors when performing different splits.
- The training set has only few entries with class 'friends' since most of the subjects had no affiliation with each other.
- In case of a trivial classifier (one that always assigns one class), the statistical methods won't work well (there will be a warning) so these cases shouldn't be taken into account.

Submitting Your Solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle.
- save your expanded .ipynb iPython notebook in your working directory.
  Submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted.
  Each student must submit her own ipynb notebook!
- we check for plagiarism. Each detected case will have the consequence of 5.0 for the whole exercise grade.
- deadline: please check slide nr 2 (this slide-set)

Citations

1. N. Eagle, A. Pentland: *Reality Mining: Sensing Complex Social Systems.* Personal and Ubiquitous Computing, 10(4), 255-268, 2006. (PDF)

2. https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees/#jp-carousel-12430