Questions 1: Learning for Pricing

**Question 1 (Customer model and demand curve).   
Describe a basic model of customers.**

There is a seller and a customer.

Basic scenario is: one seller(monopoly) which has an infinite

Amount of goods and can decide the unique price of it. Every customer has its own evaluation over the good. If price is less than, will buy, otherwise not buy.

Oligopoly: small number of sellers competing each other (Nash equilibriums)

Competitive market: There are a lot of sellers ->Arrow-Debreu equilibrium.

P price

Q(p) demand at price p

C(q(p)) cost of q(p) items for the seller

P \* q(p) revenue at price p

P\*q(p) – c(q(p)) profit at price p

Seller is a profit maximizer

**Discuss what the demand curve is.**

Demand: y axis, number of customers that would buy for the given price

Price: x axis

In real world it’s very difficult to know the real demand curve.

Marginal Revenue(q)

Marginal Cost(q)

Best Price : price p\* where MR = MC

Best demand quantity q\* = q(p\*)

Optimal social price: Price where profit of the seller is 0.

Deadweight loss of monopoly : every item is sold at same price. Efficiency improves if price can be set based to the class each customers belongs to.Disaggregate demand curves.

**Discuss what the demand elasticity is.**

Elasticity: derivative dD / dP (P) \* P / D. corresponds to normalization of derivative of demand respect to price.

Conversion Rate: probability that user buy product given that he has visited the page with his price.

**Discuss the relationship between demand curve and strategic substitutes and strategic complements.**

Good A/B can substitute good B/A. B and A are similar. If A/B reduce price, also the other need to reduce price.

A/B can complement good B/A.

Independent goods must be priced together to maximize joint profit.

**Question 2 (Pricing with known demand curve)  
Provide the mathematical model of the oligopoly scenario.**

**Describe the goal of the pricer and the mathematical conditions when that goal is achieved.**

**Given a demand curve p(q), MC(q), MR(q), find the best price and identify, graphically, the corresponding revenue and the profit.**

**Describe the deadweight loss of monopoly.**

See above.

**Question 3 (Aggregate demand curve and price discrimination)  
Discuss mathematically how different users can behave differently in a pricing scenario.**

**Discuss the difference between aggregate and disaggregate demand curves.**

**Discuss which price disaggregation is and why it is useful.**

**Question 4 (Learning setup for pricing).  
Describe which parameters need to be learnt in pricing scenarios.**

**Provide the basic learning model in pricing scenarios.**

**Which is the nature of this learning problem.**

**Question 5 (A{B{k testing).  
Describe, technically, what an A{B{k test is.**

Technique to learn the best prices. we have a set of candidates(prices), a group of users (50% production, 50% test, A/B). Prices are assigned to A or B.

Profit Maximization: Demand to estimate (Gaussian distribution for every price), Conversion Rate to estimate(Binomial dist. For every price), Marginal Cost to estimate (Gaussian).

Volumes Maximization: Demand to estimate, Conversion Rate to estimate.

Given candidates c1 and c2, collect samples for each one. So we have mean and confidence interval. Collect sufficient number of samples so that confidence intervals are separated, so we can see what candidate is the best, and exclude the worst. So we run again the test with the winner and another candidate.

Different approach: evaluate all candidates simultaneously: A/B/n testing

Collect samples for all candidates within a time horizon. Then we select the winning candidate, and use it. Two phases: Exploration and Exploitation. The expected reward depends by the length of the two phases. The ratio must be balanced to explore but not too much to not lose reward.

**Question 6 (A{B{k testing properties).  
Describe, informally, what an A{B{k test is.**

**Discuss the main drawbacks of A{B{k testing.**

Assumption of stationary process. If we discard a candidate it is never reconsidered. It is inefficient because in future it could be good.

Long time to identify optimal candidate. May not find it. May discard a potentially optimal candidate. Low confidence allows to decide the winner in short time, but this could not be optimal.

**How A{B{k testing is related to exploration/exploitation strategies.**

**Question 7 (Bandit algorithms and A{B{k testing).  
Discuss what a bandit algorithm is.**

**Discuss the main differences between bandit algorithms and A{B{k testing.**

**How bandit algorithms are related to exploration/exploitation strategies.**

**Question 8 (Regret).  
Discuss what the regret is and why, in online settings, regret is a meaningful performance index.**

**Discuss the main differences between bandit algorithms and A{B{k testing.**

**How bandit algorithms are related to exploration/exploitation strategies.**

**Discuss the difference between reward maximization and regret minimization.**

**Question 9 (Bandit formal framework).  
Describe the formal framework of a bandit problem.**

**Discuss the functioning of the UCB1 algorithm.**

**Discuss the functioning of the TS algorithm.**

**Discuss how the regret bounds of UCB1 and TS asymptotically depend on the time horizon and on the number of arms.**

**Question 10 (UCB1).  
Given three arms with Bernoulli outcomes and their clairvoyant realizations, apply the UCB1 algorithm for 5 time points.**

**Question 11 (Extending the basic bandit framework).  
Discuss the bandit model with delayed feedbacks.**

**Discuss how delayed feedbacks affect the regret bounds of the UCB1 and TS algorithms.**

**Discuss the relationship between regret and number of arms with continuous-arm problems.**

**Discuss with which conditions an optimal number of arms can be provided with continuous-arm problems.**

**Question 12 (Non-stationary bandit settings).  
Discuss why assumptions are needed when dealing with non-stationary bandit settings.**

**Describe the main assumptions for non-stationary bandit settings.**

**Discuss how basic bandit algorithms can be modified to deal with non-stationary settings.**

**Discuss how the regret bounds change in non-stationary environments.**

**Question 13 (Context generation).  
Discuss the relationship between context generation and bandit algorithms.**

**Discuss why context generation can be useful.**

**Provide a formal framework for contexts.**

**Provide a greedy algorithm for context generation**

Questions 2: Learning for Matching

**Question 1 (Matching problem).  
Provide the mathematical formulation of a matching problem.**

Start from a graph. A matching is a subset of edges of a graph such that no pair of edges share the same vertices. Bipartite graphs.

Maximal matching: we cannot enlarge it.

Maximum matching: matching with largest cardinality.

Perfect matching: all the nodes are in the matching.

**Describe what alternating and augmenting paths are.**

Alternating path is a path (every node occurs no more than once) such that edges in the matching and edges of the complementary alternates in the path.

Augmenting path: an alternating path starting and ending in unmatched vertices.

The complementary of an augmenting path is an alternating path, whose cardinality is strictly larger.

Matching is maximum if it does not admit an augmenting path.

Algorithm: look for augmenting path, make the complementary, until no augmenting path is present. See slides.

**Describe the functioning of the alternating-path algorithms for bipartite graphs.**

Slides 2-02

1. Consider a bipartite graph.
2. For the given matching find unexplored nodes and start breadth-first search. (here E, J)

A picture containing diagram

Description automatically generated

1. Start from one of them and label as even
2. Then expand the node to the one it has edge with and then expand this node 1 step further, since the path must be alternating.
3. Add the last node to the list of unexplored nodes. (Here we explored E, and added A to unexplored)

Diagram

Description automatically generated

1. Once you expanded an unexlored node, mark it as expanded and move on to the next unexlored node.
2. Repeat steps 2-6 until your node will lead you to a node that has already labeled as even (blue). This signals that you found an augmented path

Diagram

Description automatically generated

1. Consider all complementary paths and choose the one that matches max edges.Diagram

   Description automatically generated

**Describe the functioning of the alternating-path algorithms for arbitrary graphs.**

1. Consider an arbitrarygraph.
2. For the given matching find unexplored nodes and start breadth-first search.
3. Start from one of them and label as even
4. Find all edges that chosen node leads to and expand, then expand these nodes 1 step further, since the path must be alternating.
5. Add the last nodes to the list of unexplored nodes. (Here we explored b, and added F, H, I to unexplored)

Diagram

Description automatically generated

1. If, expanding an even node, the even node is directly connected to a vertex previously labeled as even belonging to same subtree, then shrink the blossom and connect the expansion of all the nodes of the blossom to the vertex of the blossom.  
   Diagram

   Description automatically generated
2. We continue to expand the blossom and, if an augmenting path is found, the blossom will be expanded in order to find the complementary pathDiagram

   Description automatically generated
3. Consider all complementary paths and choose the one that matches max edges  
   Diagram

   Description automatically generated

**Question 2 (Alternating-path algorithm for bipartite graphs).  
Given a bipartite graph and a matching, apply the alternating-path algorithm to find all shortest augmenting paths, if any.**

**Question 3 (Alternating-path algorithm for arbitrary graphs).  
Given an arbitrary graph and a matching, apply the alternating-path algorithm to find all shortest augmenting paths, if any.**

**Question 4 (Combinatorial bandits).  
Define what a combinatorial bandit problem is.**

Combinatorial bandits: can pull a set of arms subject to combinatorial constraints.

Arm: candidate

Superarm: set of candidates

Feasible superarm: satisfying the constraints

Reward: sum of the arms reward, maximize cumulative expected reward.

Constraints: Knapsack, Independent set, Matching.

**Describe how a matching problem can be formulates as a combinatorial bandit problem.**

In a Weighted graph.

Using Thompson Sampling we draw a sample for each edge, and find the best matching, then collect feedbacks, update beta distributions, repeat…

**Describe the Combinatorial Thompson Sampling algorithm.**

Initially, for **every arm**, be careful arm not super arm, we draw a sample according to the Beta distribution associated with the arm. Then, we play the **super arm** with the maximum reward. Finally, we update the Beta distributions accordingly. So, basically, the functioning is the same of the basic Thompson Sampling except that, in order to find the super arm with the maximum expected reward, we need to solve a combinatorial problem. (2-04-9)

**Define the regret in the case of combinatorial bandit problem and discuss the differences between the regret bounds of the non-combinatorial and combinatorial cases.  
Graphical user interface, text, application, chat or text message

Description automatically generated**

The regret keeps to be essentially the same of the basic version of the Thompson Sampling algorithm. So, it is logarithmic in time, while it is linear in the number of arms. Consider that in many cases we do not solve the combinatorial problem exactly since it is an NP-hard problem, but an approximation algorithm is used. This introduces an additional regret term.   
Regret depends exponentially on the number of super arms, but the application isn’t practical, because the number of feasible solutions is exponential.

**Question 5 (Online matching).  
Define what an online matching problem is and how it distinguishes from the non-online case.**

Non online: graph is completely known a priori.

Online: graph and weights are discovered during time.

Bipartite graph: left side is known, the other is unknown initially. At each round, a single node enters and its edges are discovered.

Online inefficiency: See example in slides. The cardinality of any maximal matching is at least ½ of the cardinality of the maximum matching.

**Define the competitive factor of an online problem.**

The competitive factor of an online algorithm is defined as the minimization over all the problems between the ration between the value provided by the online algorithm and the optimal clairvoyant algorithm. Where the clairvoyant optimal solution is the best solution if we had the complete information of the problem.

**Graphical user interface

Description automatically generated with low confidence  
Show that a basic online matching problem (in bipartite graphs, with the nodes of only side entering dynamically) does not admit any deterministic algorithm with competitive factor larger than 1/2.**

In the online bipartite graph matching problem, in which the left-side nodes are known beforehand and the right-side node enters one by one, the best possible competitive factor of a deterministic algorithm is 1/2. The algorithm is very simple. Match a node with an arbitrary unmatched node. The proof follows from two steps. The first step is that the cardinality of any maximal matching is at least 1/2 of the cardinality of the maximum matching. The second step is that the previous algorithm always finds a maximal matching.

We briefly sketch the proof of the first step.

1. Suppose to have a maximum matching M\* and a maximal matching M.
2. For every edge e in M\*, at least one node of the edge e is in the maximal matching M, otherwise M is not maximal. Indeed, we could add edge e to M.
3. So, the number of nodes in M cannot be smaller than the number of edges in M\*.
4. Obviously, the number of matched nodes in M\* is 2 times the number of edges of M\*
5. And so, the number of edges in M is not smaller than the number of edges of M\* divided by 2.

The second step of the proof shows that we always find a maximal matching. Suppose that M is maximal. For every edge e given by nodes (u,v) consider the round in which v arrives. We have that either v has been matched with some node other than u or v has not been matched since all the nodes (u included) were already matched. In both cases, the edge cannot be added. Notice that we cannot have a case different from the third and the fourth. (2-05-21)

**Describe a greedy algorithm for a basic online matching problem with competitive factor 1 - 1/e**

Generate randomly an ordering over the nodes that are initially available (left side)

Match a node with the first unmatched node in the ordering

This algorithm has a competitive ratio of 1-1/e

**Question 7 (General online matching).  
Show that when both sides of a bipartite matching problem enter dynamically there is not online algorithm with strictly positive competitive factor.**

No deterministic competitive algorithm even when all nodes stay the same number of rounds in the problem (when k is the same for all the nodes). The proof is a simple counter example.

1. Suppose that all the nodes stay in the problem 2 rounds. Suppose that node A enters the problem.
2. Suppose that at round one node B enters the problem and it can be matched with node A. The weight of the matching is 1.
3. We must decide whether matching A and B or not.
   1. Assume that we match them. Assume that node C enters the problem and the edge between B and C is a very high value.  
      So, the competitive factor, that is the ration between the online solution, one in this case, and the optimal clairvoyant solution approaches zero.
   2. Assume instead that we do not match A and B, and that, in this case, no further node enters the problem at round two.  
      So, in this case the competitive factor is 0 divided by 1, that is 0. So there is not any deterministic algorithm returning a strictly positive competitive factor. (2-06-10)

Shape

Description automatically generated

**Describe the functioning of the Postponed Dynamic Deferred Acceptance and report an example.**

This algorithm has an expected competitive factor of ¼ when the waiting time of every node is the same. When node enters the problem, generate 2 fictitious nodes (seller and buyer, 2 side market). So there is a fictitious bipartite graph. Find best matching in this bipartite graph. Assuming no node must leave the problem at this step we can move on and introduce new node. (2-06-11)

Once a node **gets its deadline** and we either match it or it will leave the problem without being matched. If a node has no label, we decide randomly with probability one half, the label, between seller and buyer.

In a case, node is already labeled. We select the contrary fictitious node associated with the node. I.e. if a node is labeled seller, we map with buyer and vise versa.

Questions 3: Learning for Advertising

**Question 1 (Introduction to advertising).**

**Describe the advertising formats.**

Search – Banner – Video – Email – Social - Others

Media: Internet, Television, Newspapers, Radio

Payment Schemes : Cost per impression / Cost per Click

**Describe how an advertising campaign is structured in sub-campaigns.**

Product – Channel – Target(Place, Time, User informations) – Economics(Bid, Daily Budget)

Ad Campaign is a table, where each row is a Sub-Campaign. Daily Budget of Campaign is the sum of all budgets of sub-campaigns.

**Describe the funnel model.**

User / Customer at top.

Awareness: User discovers the product

Consideration: The user compares the product with others

Decision: User looks for additional information

Purchase: User buy the product

**Question 2 (Pay-per-click advertising). Search advertising. Social Advertising.**

**Contextual Advertising.  
Describe the pay-per-click advertising scenario (roles and functioning).**

Advertiser -> Web Agency -> Ad Server -> Publisher -> User/Customer

Advertiser set the Bid, Budget, Target and all the infos. The User makes a query on the search engine. The search engine displays query results and related ads.

Publisher Problem: Must Produce web page including content and ads, and also define the pay per click payments. Can display ads in different positions on the page (Slots).

**Describe the formal model of pay-per-click advertising.**

Optimization Problem. Maximize /\s \* Qa \* Va

/\s = prob that user observes the slot s ->Estimated by search engine

Qa = click prob given the ad has been observed -> Estimated by search engine(Bandit)

Va = value per click -> Private info of advertiser, which communicates a bid to search engine.

/\s \* Qa = CTR Click Trough Rate

Maximize Sum over a : /\s(a) \* Qa \* Va s(a) slot in which ad a is displayed

Users follows a cascade model: Observes the first slot, then the second, and so on.

For each slot there is a probability that user continues to observe the next slot:

/\s(i +1) / /\s(i)

The problem is to find the best allocation: argmax(a) {Sum over a : /\s(a) \* Qa \* Va}

1. Sort ads in decreasing order Qa\*Va
2. Allocate according to such order

**Describe the main mechanisms of pay-per-click advertising and their properties.**

Auction: is done every time the search engine generates a web page. It defines the allocation of the ads and the pay per click payments.

Every advertiser makes a bid (max money that he would pay for a click).

Then allocation and payments are chosen by auctioneer. Allocation is found as discussed before (Using bids and algorithm described before).

Payments: GSP or VCG

GSP : Generalized Second Price

Pa = Qa+1 / Qa \* Va+1 (Ads sorted in decreasing order)

Used by search engines.

VCG: see slides for formulas. Bidding the true value is the optimal strategy of every player.

Used by social networks and contextual advertising.

Repeated Acutions: The previous scenario Is repeated until the daily budget of advertisers is not expired.

The bid for a sub campaing can be changed during the day, while the budget can be set only once per day.

Optimization Problem: Given a daily budget for a sub campaign, what is the best (constant) bid???

**Question 3 (Display advertising).   
Describe the display advertising scenario (roles and functioning).**

User/customer – Ad Exchange – Ad network - Advertisers

1. Advertisers communicates their ads to the ad network (bids, budget)
2. User visit a web page of the publisher
3. Publisher contacts exchange providing infos on the web page to generate(user, context)
4. Ad exchange contacts ad networks providing the infos
5. Every ad networks runs auctions to determine winning ad
6. Every ad network reply
7. Ad exchange runs auction and communicates the winner to the publisher
8. Publisher generate the web page

Every ad network runs a second price auction to determine the winner of the network and its payment. Ad exchange runs another auction. Ad networks communicates winning bid and the ad, and also another optional bid (optional second price). See slide for example.

**Describe the auction mechanism used for display advertising.**

See above.

**Question 4 (Optimization problem).  
Provide the mathematical formulation of the advertising optimization problem.**

Assumption 1: Performance of every sub campaign independent of performance of other sub campaigns. (not true in general).

Assumption 2: The values of bid and daily budget are finite and given.

See slides for optimization problem formulas.

**Describe the algorithm to solve the advertising optimization problem.**

Represent values of each sub campaign by means of a table. (3-04)

Rows: values of daily budget -> Y

Columns: values of the bids -> X

In each slot: v \* n(x, y) value of sub campaign

n(x, y) : Number of clicks given a bid and a daily budget.

1. remove dependency on the bids. (find max for each bid, for every value of daily budget)
2. new table (rows: sub campaign, columns: daily budget)

in this table some values are missing (-infinite)

1. new empty table to fill:

0 10 20 30 40 50 60 70

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| +c1 | -inf | 90 | 100 | 105 | 110 | -inf | -inf | -inf |
| +c2 | Max{-inf} | Max{-inf, 90} | Max{-inf, |  |  |  |  |  |
| +c3 |  |  |  |  |  |  |  |  |
| +c4 |  |  |  |  |  |  |  |  |
| +c5 |  |  |  |  |  |  |  |  |

Temporary row (From second table):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **C1** | -inf | **90** | **100** | **105** | **110** | **-inf** | **-inf** | **-inf** |
| **C2** | 0 | **82** | **90** | **92** | **-inf** | **-inf** | **-inf** | **-inf** |

**…**

Find value of best allocation for each cell. Maximization of some terms. Max{

Non si capisce un cazzo.

**Discuss the complexity of the algorithm.**

O(N \* K^2)

N: # of sub campaigns

K: # of values of the daily budget

the complexity of the algorithm is linear in the number of sub campaigns and quadratic in the number of values for daily budget. So, the algorithm is computable in real time unless we have a huge number of daily budget values. (3-04)

**Question 5 (User model and regression).   
Discuss why a structure behind the user model needs to be assumed in practice.**

**(?)**

Text, letter

Description automatically generated

The optimization problem can be solved once we know the values of the function n\_j for every pair (x\_j, y\_j). Since we have finite set of pairs, what we need is to estimate the value of that function for every pair. However, learning every possible combination of values is impractical, as they would require too many datas. For instance, 10 values of bid and 10 values of budget generate 100 different combinations. Collecting a sample for every pair would require 100 days. 10 samples per pair would require 1000 days. We need to adopt a different approach. First we assume that the function is smooth and that the dependency of the function on bid and budget is factorised. **Describe a user model.**

**(?)**

Suppose to fix the bid to a constant and to represent the curve between the number of clicks and the daily budget. The function is equal to zero if the bid is too small, since the advertiser will not take part in any auction.

on average the dependency on the budget is initially linear and then the curve keeps to be a constant since there is no other auction in which being displayed.

Furthermore, the function is zero whenever the budget is smaller than the bid.

Given two bids, x and x prime. We expect that the curve with x prime has a smaller slope, but it can reach larger values of n as the budget increases. The idea is that increasing the bid, the number of auctions in which the ad can be displayed increases.

**Chart, line chart

Description automatically generated  
Discuss how Gaussian Processes can be used for regression and why they are crucial for online learning.**

We have very mild assumptions on the function to estimate. GP returns a gaussian probability distribution over the outcome. It is crucial to assure convergence of bandit algorithms.

Gaussian Processes allow to tackle a regression with very mild assumptions on the function to approximate and, more importantly, returning a probability distribution over the outcome. That is, the gaussian processes also return a measure of the uncertainty of the result of the regression. A GP receives in input the past observations, the kernel of the GP and its parameters, and the point to evaluate. The output is a probability distribution over the outcome.

**Question 6 (Combinatorial bandits and advertising).  
Describe a combinatorial bandit problem.** 3-06-03

Novelty respect to classic bandit:

-Arms are correlated

-Reward of an arm provides information on the reward of arms close to it.

**Describe a Gaussian-Process bandit problem.**

See slides 3-06-04

At every time t, for every arm, we sample a reward according to the probability distribution returned by the GP. Then, we select the best arm. Finally, we update the GP according to the observed reward.

Gaussian process returns the expected value plus the confidence interval (Uncertainty). This is fundamental to assure exploration of the algorithm, and the exploration allows to reach the optimal solution. **Describe a combinatorial Gaussian-Process bandit problem. (3-06-11)**

We can pull any set of arms satisfying some combinatorial constraints.

**Describe how combinatorial Gaussian-Process bandits need to be modified to solve an advertising optimization problem. (3-06-15)**

Here, every sub campaign is modelled by means of a GP with a constraint forcing that at most a pair bid budget can be choses. So we have a number of GP equal to the number of sub campaigns. Then we have an additional constraint that is the classical knapsack constraint, forcing the cumulative budget to be not larger than a given value. So, for every sub campaign, we have our GP.

We sample for every pair bid budget the reward and then we solve the combinatorial optimization.

The dependency of the regret bound is similar to the previous ones except that in the regret we have a dependency on the number of GP, denoted with the letter C, and the information gains must be summed.

**Text

Description automatically generated**

Questions 4: Learning for Social Influence

**Question 1 (Information cascades).   
Provide the model of information cascade.**

Set of users. They can make one action among a given set. Do not have info on which is the best, initially are equally valuable. First user receives a signal (1 better than 0, with probability p). Every user can see the choice of the previous users, and they will be affected by their decisions. If all the previous users select for example 1, this nullifies the signal and every user after will choose 1.

**s  
Show that, asymptotically, the probability that an information cascade happens is one.**

If the difference between 0 and 1 choices in the past is less than 2, every user makes a choice based on the signal he received. When the difference is greater or equal 2, the signal is no more important, the cascade is activated. Actions depends no more on the priors.

Prob of choose 1: q

Prob of choose 0: 1-q

q^3 + (1-q)^3 = P 3 consecutive 1 or 0 signals.

Prob of having a cascade = 1 – (1-P)^(T/3) -> 1 the probability tends to 1 as the number of users tends to infinity.

**Discuss the mathematics behind information cascades.**

**Question 2 (Belief update in information cascades).   
Given P[1 > 0], P[ {1} | 1 > 0 ], P[ {0} | 1 > 0 ], calculate P[1 > 0 | A] for some A.**

**Question 3 (Direct effects).  
Describe the independent cascade model.**

Social network as a graph. Users and connections. Direct edges. Each edge has a parameter (prob that node A affects B). Every node can be of different states:

Susceptible -> can be activated

Active

Inactive

At beginning, a subset of nodes are seeds. For every possible edge, we report the edges that are activated -> live edge graph. Every node reached in the live edged graph will be activated.

**Describe the linear threshold model.**

Also nodes have a parameter, theta. A node activate if the sum of the edges values coming in that node is greater than theta.

**Discuss what a live-edge graph is.**

**Question 4 (Social influence in the cascade model).   
Provide a formal model for the influence maximization problem.**

Input: Network, Probabilities, Budget(Number of nodes we can buy)

Actions: subset of nodes(seeds) to select simultaneously

Goal: Maximize expected number of active nodes at the end of the cascade.

Algorithm:

E: set of edges

X is a binary vector, Xi = 1 if edge i is present, 0 otherwise

Given an X, we select a subset of edges.

Enumerate every X (there are 2^n) n num of nodes

For every X check whether there is a path connecting a node with some seed, for every possible node.

For every X compute corresponding probability.

Computation of activation prob associated to each node is impractical, exponential time in the number of edges. -> Monte Carlo Sampling.

1. assign every node Zi = 0
2. generate a live edge graph according to probability of each edge.
3. For every active node (breadth first search) in that live edge, assign Zi = Zi + 1
4. Repeat k times
5. For every node return Zi / k

How many repetitions??

See slides.

**Describe an exact algorithm to compute the expected number of nodes influenced by a set of seeds.**

**Describe an approximate algorithm to compute the expected number of nodes influenced by a set of seeds.**

**Provide a theoretical bound of the approximate algorithm.**

See slides

**Question 5 (Influence maximization in the cascade model).  
Describe the exact algorithm to maximize the social influence given a budget constraint.**

**Describe an approximation algorithm to maximize the social influence given a budget constraint.**

**Discuss the theoretical guarantee of this approximation algorithm.**

**Question 6 (Learning and influence maximization).  
Discuss what the main learning issues are in influence maximization and provide a learning algorithm for each of these.**

Uncertainty: we know the graph but we don’t know the probabilities. Some information is not present. We repeat the problem in time. Exploration to estimate the probabilities.

3 scenarios:

1. we can observe the activation of all the edges. (like retweet in twitter).
2. We can observe activation of a small portion of the edges.
3. We can observe only the activation of the nodes, but not information on the edges.

See slides for algorithms.