Multi Armed Bandit

- abbiamo una collezione di k advertisements che possono essere visualizzate nel nostro sito. Ogni volta che un user entra nel sito gli mostriamo un ad. Ogni visualizzazione è detta impression. Se l’user o visita si trasforma in click

- obbiettivo: definire una policy per selezionare ad ogni round l’ad migliore, in modo da massimizzare i clicks.

- Tentativo Naive: provare alternativamente le ad fino a quando una di queste ottiene un click. Da li in poi selezionare sempre quella. Sbagliata perché da una decisione prematura. Abbiamo bisogno di più evidenze.

- EXP-EXP: Introduciamo un algoritmo con una prima fase di ESPLORAZIONE e una seconda di EXPLOITATION. Il parametro di tale algoritmo viene definito come N. Se N ha un valore piccolo abbiamo il rischio di fare una decisione prematura (little exloration, large exploitation), se N ha un valore grande abbiamo una grande chance di scegliere il vero vincitore, ma sarà molto lento e noi volgiamo trovare il vincitore il prima possibile. N dipende dalla natura degli annunci. Se uno performa molto meglio dell’altro, il costo di esplorazione che dobbiamo sostenere sarà piu basso. Se sono simili, abbiamo bisogno di molta esplorazione e questo implica costi maggiori

- L’esplorazione implica il sostenimento di costi per la scelta di un ad che non sia la migliore. Viene inteso come un investimento necessario per l’acquisizione di conoscenza.

- GREEDY policy: rendere una policy Greedy significa renderla focalizzata sull’ad che da risultati migliori. Al round t, l’ad che verrà scelta dipenderà dalla storia dell’intero gioco fino al round t-1. Quello che faremo sarà in questo caso provare ogni ad una volta (è l’errore!) e da li in poi scegliere l’ad migliore fino a quel momento. Se le ads ad un certo round si equivalgono, la prossima viene scelta random.

La soluzione non sarà del tutto soddisfacente. Infatti, se ci troviamo in una situazione in cui abbiamo 2 ads, a e b, a con 1 impressione e 1 click e b con 0 click e 1 impressione, anche se a non viene cliccata ad ogni round, il suo success rate decrescerà ma non raggiungerà mai 0. Quindi rate(a) > rate(b) per sempre. Quello che facciamo sarà allora estendere la fase di esplorazione dicendo: proviamo un ad non una volta ma N volte e poi proviamo l’ad che performa meglio fino ad adesso. Abbiamo sempre però il problema di scegliere il giusto investimento nell’esplorazione.

Quello che facciamo sarà mettere insieme le fasi di esplorazione e exploitation usando dati provenienti dal processo stesso di selezione.

- EPSILON-GREEDY policy: è un mix tra la policy greedy e una policy random (ad ogni round seleziona un ad a caso). CI dice di scegliere un valore per epsilon che sintonizza adattivamente il tasso di esplorazione, e poi provare ogni ad una volta e dopo scegliere la miglior ad in termini di performance fino ad adesso con probabilità 1 – epsilon o scegliere l’ad in modo random con probabilità epsilon.

Quello che fa è minimizzare la fase iniziale di esplorazione e poi alternare exloration e exploitation.

Per esempio: scegliamo epsilon = 0.1 = 10%. 2 annunci a e b, il leader b viene selezionato con una probabilità di 90% + (10%/2) , a viene selezionato con una probabilità del 5%. Se erano 4 annunci il leader aveva una prob del 92.5% e i 3 followers 2.5%.

Il problema principale è scegliere il parametro Epsilon. Questo è difficile e dipende da cosa accade durante il gioco.

Epsilon serve per sintonizzare adattivamente il tasso di esplorazione e ha bisogno di una sintonizzazione adattiva anche per se stesso

- Per raggiungere il TUNING DI EPSILON, il modo più comune è definire uno SCHEDULING, una policy che assegna a epsilon un valore che dipende dal tempo e che solitamente decresce con il tempo. Posso avere uno scheduling che non decresce sempre nel tempo perché può succedere che cambi il Ctr e allora voglio cambiare il tasso di esplorazione

Questo perché l’exploration rate deve essere decrescente nel tempo per due motivi:

con più tempo (più rounds), la mia conoscenza sarà maggiore e avrò bisogno di esplorare meno. Con meno tempo nel futuro ho meno possibilità di fare exploiting della conoscenza che ho accumulato

Un possibile MIGLIORAMENTO può essere connettere il valore di epsilon non solo al tempo ma anche all’incertezza (t = tempo, u = incertezza). Epsilon sarà tipo u / t.

Incertezza sarà grande se il leader ha una small edge rispetto ai competitors, quindi non siamo sicuri che il leader sia realmente il migliore oppure se le ads cambiano il loro true success rate (true positive)

Benefici: permette di definire il giusto investimento in esplorazione in modo più accurato rispetto alla policy Greedy.

Svantaggi: è difficile fare il tuning di epsilon, è un limited ranking, cioè la probabilità di un ad di essere selezionata dipende dal suo rank. Non è una scoring technique come softmax, in cui il punteggio è trasformato in una probabilità di essere selezionata.

- META LEVEL EPSILON GREEDY : quello che facciamo è selezionare un set di possibili valori di epsilon. Un meta parametro epsilon viene usatto per dare regole alla competizione tra questi valori e ad ogni passo selezioniamo un valore di epsilon e applichiamo epsilon greedy con quel parametro. Quindi, il meta parametro sintonizza il tasso di esplorazione per trovare il miglior parametro mentre gli altri parametri fanno lo stesso per le ads

- MULTIPLE WINNER EPSILON GREEDY : se vogliamo mostrare k < K ads, possiamo applicare epsilon greedy k volte. Selezioniamo il primo vincitore e dopo eseguiamo epsilon greedy su k-1 ads, escludendo il vincitore. Ripetiamo questo fino a che k ads sono selezionate

- BATCH EPSILON GREEDY : nella realtà gli users che agiscono sul nostro sito sono molti e non abbiamo abbastanza tempo per fare una selezione e mostrare l’ad selezionata. Quindi, un round viene inteso come un intervallo di tempo in cui molte decisioni verranno fatte usando la stessa policy e gli stessi dati. Durante un round non possiamo fare update dei dati, quindi la 1000 esima ad sarà selezionata senza imparere dalle precedenti 999 decisioni. Solo dopo che il round finiscere possiamo aggiornare i dati per rendere il prossimo round più informato.

- SOFTMAX : obbiettivo è fare uno scoring delle ads e capire quanto un ad è migliore rispetto all’altra. L’idea chiave è rendere a piu propensa ad essere selezionata rispetto a b se a è meglio di b.

epsilon Greedy fa ad ogni round una scelta tra due policy. Esplorazione e exploitation sono mixati ma rimangono visibilmente distinti, o esplorazione o exploitation. Softmax invece mixa continuamente ad ogni round le due fasi. L’idea è scegliere la max performance ma in un modo soft, non ogni volta ma spesso.

L’algoritmo dice che per ogni round, calcolare la qualità dell’ads (il Ctr), calcolare la probabilità e selezionare l’add in accordo con la probabilità. Dopo fare update dei dati (impressions, click, ctr).

Fondamentalmente, la probabilità è un rapporto tra valori esponenziali divisi per la temperatura.

Senza la funzione esponenziale, la versione dell’algoritmo sarebbe troppo naive. a è 3 volte meglio di b, allora ha probabilità 3 volte piu grande di essere scelto, che è lo stesso di avere epsilon greedy con epsilon = 25%. Allora introduciamo la funzione esponenziale

Si usa la funzione esponenziale perché cosi si esprime un processo di crescita nel quale il growth rate è proportionale alla current size dell’add ( se a è 3 volte meglio di b, a ha la probabilità di essere selezionata non 3 volte piu grande, ma e^3 = 20 volte piu grande.

Senza la funzione esponenziale l’algoritmo sarebbe troppo meritocratico, nel senso che se a è 2 volte meglio dell’altra avrà possibilta moltiplicata per 2 di essere selezionata

L’esponenziale AMPLIFICA LE DIFFERENZE, quella migliore viene presa troppo piu spesso rispetto alle altre, pero siccome questo rende l’algoritmo troppo greedy allora uso la temperatura

La trasformazione esponenziale esalta le differenze dando all’ad migliore una probabilità di essere scelta più che proporzionale, SFRUTTO LE MIGLIORI Ads SENZA ESSERE GREEDY. Essendoci un rank esponenziale, le altre ad non hanno valore 0.

La TEMPERATURA, sintonizza il tasso di esploazione (un meccanismo adattivo che fa cambiare il tasso di esplorazione nel tempo), piu grande è la temperatura e più alto è il tasso di esplorazione.

Temperatura alta -> esponente vicino a 0 -> entrambe gli esponenziali vicini a 1 e molto simili tra di loro. Alta temperatura significa poca attenzione alle differenze

Temperatura bassa -> esponenti alti -> entrambe gli esponenziali alti, ma qualità alte sono piu esaltate delle qualità basse. Bassa temperatura significa grande attenzione alle differenze.

La migliore ad ha una probabilità assegnata vicino a 1, tutte le altre hanno una probabilità vicina a 0.

Grazie alla temperatua: partiamo con una t vicina a 1. Le differenze sono minime e in pratica softmax approssima la policy random. Il rate di esplorazione è vicino a 1. Poi gradualmente riduciamo T (SIMULATED ANNELING). Le differenze nella qualità si fanno sempre più importanti e il tasso di esplorazione decresce. Continuiamo a ridurre la temperatura fino a farla arrivare vicino a 0. Progressivamente softmax apporssima la greedy policy. Con il passare del tempo softmax si muove da quasi random a quasi greedy.

- THOMPSON SAMPLING: la sua idea principale è che la probabilità di un ad di essere selezionata è uguale alla probabilità di essere la migliore. (se a è 2 volte più probabile che sia migliore di b, ha il doppio di probabilità di essere selezionata)

Nel lungo periodo Thomposon sampling converge a soluzioni ottime. E’ self-adaptive, cioè nuove osservazioni cambiano progressivamente la probabilità che l’ad sia selezionata. Impara in un framework BAYESIANO. In un framework bayesiano parto con dei pregiudizi, e poi mano a mano che ho delle evidenze ottengo piu conoscenza e aggiusto la curva.

Non richiede un paramentro come l’exploration rate in epsilon greedy o la temperatura in softmax. La conoscenza iniziale che ho deriva dallo storico dei dati, si puo partire con annunci che hanno la stessa probabilità di essere selezionati oppure no, non è essenziale. Questo perche potrei sapere qualcosa da campagne precedenti.

Invece di usare l’observed ctr, creo due distribuzioni beta e da li estraggo un valore random come il mio ctr. Questo perche io non conosco il vero ctr. Il ctr è una rappresentazione randomica di un ctr vero che non possiamo sapere, perche senno sapremmo esattamente quale ad mostrare.

Dato il Ctr creaiamo una distribuzione beta per ogni prob. Per ogni distribuzione tiriamo fuori un valore random. Li copariamo e scegliamo l’ad che ha valore random piu alto. Rappresento, Facciamo update di hints o fallimenti.

Differenza se parto da beta (1,1) o beta(1000,1000), impiegherò di piu a far variare la distribuzione perche un click in piu avrà meno peso.

Bayesian learning

Il teorema di Bayes è uno strumento principale per la previsione (forecasting), ci aiuta a modificare la stima delle probabilità in una maniera razionale, sfruttando conoscenza pregressa a priori, calcolando conoscenza a posteriori. Ci aiuta a insistere sulla campagna che ha più probabilità di essere cliccata.

Data un’ipotesi e un evidenza:

![Immagine che contiene testo

Descrizione generata automaticamente]()

P(H|E) è la probabilità a posteriori

P(E|H) o likelihood è la probabilità di osservare E in un mondo dove le nostre ipotesi sono vere

P(H) o prior probability è la nostra confidenza nell’ipotesi H, prima di acquisire nuove informazioni

P(E) è la probabilità di osservare l’evidenza in una collezione di mondi possibili in cui l’ipotesi è vera o falsa.

Il rapporto ha un valore >1 se osservare l’evidenza è più probabile quando l’ipotesi è vera.

Il rapporto tra likelihood e probabilità dell’evidenza è detto LEARNING FACTOR e ha valore 1 quando la probabilità di osservare l’evidenza è lasciata invariata dall’ipotesi. Allora H e E sono indipendenti

Naive Bayes Classifiers

Si vuole scegliere l’ipotesi che massimizza la probabilità a posteriori.

Il problema è la likelihook.

Assumiamo l’ipotesi di indipendenza, la conoscenza su uno degli attributi non ci serve per stimare gli altri. Questo vuol dire che il calcolo della likelihood viene fatto come calcolo di più likelihood separate nel caso l’evidenza abbia più attributi (come nell’esempio Magazine = yes, watch = yes, life = no, credit = no), quello che si fa è calcolare la likelihood dei singoli attributi, non delle loro combinazioni. Non devo fare il calcolo di tutte le combinazioni

Questa assunzione viene fatta in modo da poter considerare un sample più ampio e posso sfruttare al meglio i dati, ho un guadagno in termini computazionali, posso lavorare anche se ci sono datasets che contengono buchi (0 imp, 0 clicks). Chi quadro serve come indicatore di indipendenza, se è 0 c’è indipendenza, piu è grande e più c’è dipedenza.

Naive Bayes Class sono MOLTO EFFICENTI, questo perchè usiamo le ipotesi solo per stimare un ranking, non stiamo cercando uno scoring che ci dice quanto un’ipotesi è meglio dell’altra.

È chiamato cosi Naive perche lo facciamo fuzionare con indipendenza degli attributi anche se non sappiamo se ci sia.

Probability distribution

Possono essere discrete (collezione finita di valori reali), infinite, continue (assumere infinit valori realI)

- Bernoulli Distribution : è un modello per un evento random con 2 possibili risultati, P(1) = p e P(0)= 1-p. La somma delle singole probabilità deve essere 1. La cosa interessante qui è che non sappiamo cosa sia p. Stiamo costrunedo un modello. Assumiano il Ctr come una quantità virtuale, qualcosa che immaginiamo come un generatore di clicks. Il Ctr osservato è diverso, è il ratio tra clicks e impressions che si è manifestato realmente

- Distribuzione Binomiale : lunga lista di eventi, ognuno modellato come un evento bernoulliano.

La curva diventa sempre più piatta muovendosi verso 0.5

- Beta Distribution: in tale distribuzione il primo parametro è il numero di hits e il secondo il numero di fallimenti. Tale distribuzione descrive la probabilità che il vero CTR cada all’interno di una certa regione.

- Normal Distribution : detta anche Gaussiana. La sua descrizione si basa su due valori, la media e la varianza. Data media e varianza possiamo sapere quanto è possibile osservare la realizzazione di un fenomeno modellato con la distribuzione che cada in un certo range

Decision theory

Procedura ricorsiva che va all’indietro dalle foglie alla radice

Essere razionale significa massimizzare l’utlità.

Cerchio = event node, il suo valore significa in media in un gioco ripetuto molte volte

Rettangolo = decision node

Prezzo di indifferenza è una trashold tra quanto è razionale pagare o meno, è la differenza tra l’utilità expected pagando per l’informazione e il gioco senza informazione visto precedentemente.

Trendlines

Una trendline è una curva che descrive la tendenza di base del fenomeno da predire

Le funzioni principali per il forecasting sono: lineare, polinomiale, esponenziale, logaritmica, power

Nella predizione assumiamo anche un certo ERRORE dovuto al rumore.

- R^2 = determinante lineare, ci dice quanto è meglio il mio modello rispetto ad utilizzare la media dei punti. R^2 = 74 vuol dire che il mio errore è ridotto del 74%. Si calcola come sommatoria delle differenze al quadrato tra valori e valori corrispondenti predetti.

Lineare = ax + b

Quadratica = ax^2 + bx + c 🡪 R^2 + meglio rispetto a quello lineare

Power = ax^b

Esponenziale = ae^bx

Logaritmica = a ln(x) + b

- OVERFITTING : quando troviamo un equazione che approssima perfettamente i data points, corriamo il rischio di imparare troppo e troppo bene quello che è successo, il rischio potrebbe essere imparare dal NOISE.

Vogliamo imparare il succo dei dati, non i dati stessi.

Tale curva di grado n, che troviamo con questa equazione, avendo n data points è detta INTERPOLAZIONE POLINOMIALE e riproduce perfettamente i datapoints.

Questa cosa va bene per predire il passato, NOI VOGLIAMO PREDIRE IL FUTURO.

Quello che dobbiamo saper gestire è un trade off: pochi dati ci datto modelli semplici ma poco accurati, molti dati ci danno modelli molto accurati ma con complessità molto alta-

Regressione

- Obbiettivo: stimare o predire valori non conosciuti. L’Idea è che se l’equazione può ricostruire con buona approssimazione il training set (sample di dati che prendiamo in considerazione), può ricostruire anche samples non conosciuti.

Nella regressione possiamo avere una compressione CON o SENZA perdita di informazione. Nella decodifica della compressione, ci posso essere delle approssimazioni dette DISTORSIONI.

Matematicamente la distorsione viene calcolata come la sommatoria delle differenze al quadrato tra il valore xi e una costante k.

Il problema adesso è trovare la cosante che minimizza la distorsione. Il valore ottimo della costante è uguale alla media dei valori che abbiamo. Questo perché possiamo assumere che ogni punto del sample è uguale alla media del sample. Valore più ottimo non si può trovare.

Se k = media allora la distorsione media è la varianza.

La media è la scelta migliore del parametro per un modello polinomiale di grado 0.

- INTERPOLAZIONE : cerca una funzione che predica valori futuri con accuratezza e minimizzi la distorsione.

Ci sono dei limiti di cui abbiamo bisogno per non incorrere nell’overfitting:

la funzione deve fittare bene i dati del training set e non deve cambiare molto se il training set cambia. Si accettano quindi delle distorsioni per non incorrere nell’overfitting.

Di solito, se il grado della regressione polinomiale aumenta, generalmente la distorsione diminuisce, ma la varianza aumenta.

Regressione logistica

- obbiettivo: predire l’andamento di una campagna pubblicitaria, un’epidemia. Predire qualcosa di non conosciuto.

Funzione logistica è formata da funzione esponenziale, funzione crescente e funzione logaritmica.

Funzione esponenziale = più grande è il valore, più cresce velocemente, più piccolo il valore, più lentamente decresce

Funzione logaritmica = inverso della funzione esponenziale, più grande il valore, più lentamente cresce, più piccolo il valore, più decresce velocemente

La funzione logistica ha y compreso tra 0 e 1 esclusi. Se x tende a 0 y tende a ½

0 è escluso perchè, anche se per adesso non ci sono cliks, non voglia interferire con il fatto che l’evento non possa capitare in futuro. Ragionamento opposto vale per 1.

E’ una funzione molto usata perché in essa c’e un consumo esponenziale limitato delle risorse.

Rappresenta molto bene situazioni reali.

Quello che dobbiamo fare è trovare dei best fitting parameters, i coefficienti beta (che devono avere un valore non maggiore di 1 e nom minore di 0)

Per quanto riguarda i coefficienti:

Beta 0 = logaritmo di un odd, ci dice qual è il giusto investimento da fare per un cliente del segmento base

Beta 1,2,3 = logaritmo di un odds ratio, non è un valore, è una variazione di un valore, ci dice l’investimento in piu che dobbiamo fare per sostituire il customer del segmento base con un customer di un segmento che differisce dalla baseline con tutti 0

Odds = giusto investimento da fare per far si che un potenziale acquirente acquisti. (p /1-p) SUCCESSI / INSUCCESSI

- Assunzione di indipendenza : Serve a confrontare il profilo base di un cliente con tutti gli altri che potrebbe avere. Se non la avessi, avrei bisogno di ulteriori coefficienti

Dynamic pricing

Abbiamo un sito di e-commerce, con un singolo prodotto. Ogni volta che un user entra nel sito abbiamo la ossibilità di mostrargli un prezzo differente

- Obbiettivo: Trovare il prezzo ottimo in modo ma massimizzare la revenue.

Revenue = Prezzo x Quantità

La tra domanda è in funzione del prezzo. Se il prezzo aumenta, la domanda diminuisce.

Se non avessimo la relazione tra prezzo e domanda potremmo risolvere tale problema com un multi armed bandit game in cui:

- i prezzi giocano i ruoli delle diverse pubblicità

- ogni prezzo è mostrato ad ogni customer per un certo periodo di tempo e valutato per la revenue che genera.

Dopo applichiamo determinate policy con exp-exp

In softmax la qualità = expected revenue.

Applicare thompson sampling significa usare il selling rate come se fosse il ctr

-Trucco dell’arbitraggio: capire a quale prezzo e quante unità vendere ai diversi segmenti.

Aumento il prezzo al segmento più sensibile al prezzo (faccio si che mi compri un’unità in meno) e abbasso il prezzo al segmento meno sensibile al prezzo (facendo si che mi compri un’unità in più)

- Model driver apporach: dividiamo i customers in segmenti differenziati per profilo. Quello che vogliamo fare è trovare la miglior pricing policy.

La revenue totale è uguale alla sommatoria delle revenue per ogni segmento di customers.

Prima di tutti analizziamo i dati storici per costruire un modello predittivo della domanda, poi definiamo un metodo per selezionare la miglior pricing policy basato sul modello predittivo.

La predizione della domanda in funzione del prezzo viene fatto con la regressione

L’ottimizzazione della price policy viene attuata con le derivate.

Se ho un solo segmento faccio la derivata di y = f(x) e la pongo uguale a 0. Ottengo cosi il prezzo ottimale.

Se abbiamo più di un segmento:

Abbiamo la r = p1d1 + p2d2, abbiamo le domande in funzione del prezzo e abbiamo limite di capacità. Dalla domanda ricavo il prezzo. Inserisco il prezzo nella revenue.

Faccio le derivate della revenue rispetto alla domanda 1 e alla domanda 2. Queste verranno definite come utilità marginali, che ci dirano quale sarà l’impatto di vendere un’unità in più sulla revenue di quel segmento.

Per ottenere il prezzo ottimo, pongo le due utilità marginali uguali tra di loro. Cosi ottengo le due domande sostituendo i valori nel capacity constraint. Sostituisco i valori delle domande nella formula del prezzo di modo da trovare i due prezzi, sostituisco tutti i valori nella revenue e trovo la miglior revenue possibile.

Quindi formalmente le condizioni di ottimalità devono essere:

- le utilità marginali devono essere in equilibrio, senno potrei applicare il trucco dell’arbitraggio, cioè allocare prodotti al segmento che ha utilità marginale più alta. Facendo cosi ottengo una utilità marginale per prodotto maggiore, però per la regole dei rendimenti decrescenti, l’utilità marginale diminuirà sempre di più fino a tornare in equilibrio

- la differenza tra la capacità e la domanda moltiplicata per l’utilità marginale di equilibrio deve essere uguale a zero.

Se pi greco uguale a 0, non ho convenienza a vendere un prodotto in più perche sono in equilibrio, NON VOGLIO VENDERE DI PIU’, se la differenza è uguale a 0, non ho più prodotti da vendere

- la domanda non deve essere negativa, perche se non lo specificassi, nel caso non avessi pià prodotti da vendere potrei avere utilità marginale negativa

- Hill Climbing Algo: algoritmo greedy, che fa una simulazione passo dopo passo, step by step e ogni volta faccio la scelta ottima. Trovo ottimi locali che non sempre sono ottimi globali.

Inizio vendendo 0 a ogni segmento e calcolo le utilità marginali. Devo decidere a quale segmento vendere, se le utlità marginali sono uguali allora scelgo random, se una ha utilità maggiore scelgo quella, questo pero, mi farà diminuire l’utilità marginale. Faccio questa cosa ITERATIVAMENTE fino a quando o ho finito la capacità oppuure le utilità marginali sono uguali a 0.

Viene dimostrata l’ipotesi che le utlità marginali decrescono davvero

Single Capacity allocation

Capacità limitata, obbiettivo è massimizzare la revenue, 2 periodi differenti, prezzo discount < full fare, vendere un biglietto oggi a un prezzo più basso o domani a un prezzo più alto, decidere giusto booking limit, protection level, Y = C- b,

tradeoff tra due rischi dilution e spoiling, il ragionamento che facciamo è di tipo marginalistico che si concretizza in un decision tree, stiamo creando criterio decisionale per decidere se aumantare booking limit o no. Per fare questo calcolo l’expected marginal revenue e la pongo > 0

Quello che ottengo è una disequazione dove il lato sx deve essere minore di quello dx

A sx ho la probabilità che la domanda nel secondo periodo ecceda il numero di biglietti che mi sono rimasti (RISCHIO DI DILUTION) . A dx ho il rapporto tra i 2 prezzi. E’ vero che questa disequazione vale? Se prezzo discounted è vicino a quello full fare: decido di aumentare booking limit anche se la probabilita nel secondo periodo che la domanda ecceda biglietti rimasti è alta. Questo perche se i prezzi sono simili, vendere primo o dopo non cambia.

Se i prezzi sono simili decido di aumentare il booking limit anche se il rischio di non avere abbastanza biglietti è alto

Se i prezzi sono differenti , per aumentare il booking limit, devo avare una probabilità che la domanda nel secondo periodo ecceda i biglietti rimasti molto bassa.

Questa analisi marginalistica, ci porta a creare un criterio decisionale su quando aumetare il booking limit

Questo metodo di aumentare il booking limit ogni volta di 1 mi porta a creare un algo hiil climbing.

Questo si ferma o perché ho booking limit = capacità totale o perchè il criterio decisionale mi dice di no.

Little wood rule serve per trovare il booking limit ottimale = valore che mi fa uguagliare i due membri.

- Many class problem: caso in cui abbiamo più di 2 periodi e classi. E’ importante sottolineare la condizione secondo cui il prezzo aumenta con il passare del tempo. Il nostro obbiettivo è massimizzare la revenue rispetto ad ogni stage.

Per fare questo usiamo un metodo detto DYNAMIC PROGRAMMING e quindi la funzione obbiettivo sarà massimizzare per ogni stage, il valore che otterremo dal numero di sedili venduti a tale stage per il loro prezzo.

Il constraint che abbiamo è che ad ogni stage, il numero di sedili che vogliamo vendere deve essere minore o uguale alla capacità disponibile.

Quella che usiamo è una proceduta ricorsiva a due stadi (quello attuale e tutti quelli successivi), in cui ad ogni passo abbiamo: una domanda, una capacità residua (cioè la capacità totale – il numero di biglietti che decidiamo di vendere agli stage precedenti) e una quantità che decidiamo di vendere (u). La revenue ad ogni passo sarà uguale a u per p.

Allo stage successivo la capacità rimanente sara uguale alla capacità che avevamo meno il numero di biglietti che decidiamo di vendere

La quantità che decidiamo di vendere ad ogni stage deve essere maggiore di 0, minore della domanda al tempo t e minore o uguale alla capacità residua.

N.B: u è deciso prima di osservare la domanda, noi assumiamo di deciderla dopo perché la scelta ottimale della variabile di controllo non richiede conoscenza della domanda

Un concetto chiave da introdurre è la VALUE FUNCTION Vt(x). Questa rappresenta l’expected revenue di avere x sedili disponibili al tempo t sotto l’assunzione che li useremo nella maniera ottima.

Riscrivo l’equazione come una somma pesata della revenue massima per ogni scenario della domanda, usando come peso la probabilità che lo scenario si verifichi.

Quello che stiamo massimizzando è un profitto + un profiutto atteso per un numero di biglietti uguale alla capacità residua – il numero di biglietti che decido di vendere nel periodo t.

Questa forma ricorsiva dell’equazione ha bisogno di un criterio di stop, che sarà profitto expected per il tempo n+1 = 0.

Tale processo va BACKWARD, quando programmiamo partiamo dall’ultimo step decisionale fino al primo, con una inversione temporale

- Optimal Policy: possiamo definire l’expected marginal revenue come la differenza tra questa con x biglietti e con 1 biglietto in meno.

Tale valore è DECRESCENTE NELLA CAPACITA’ E NEL TEMPO, poiché il biglietto n+1 corre un rischio maggiore di non essere venduto e più tempo passa e meno opzioni ho per la mia strategia che diventa meno effettiva

- Optimal Protection Level : massimo numero di biglietti per cui l’expected marginal revenue è maggiore rispetto alla revenue che posso ottenere oggi.