Machine Learning

Supervised learning

- Inductive bias
- Spazio delle ipotesi H con ipotesi $h \in H$
- version space, spazio delle ipotesi tra gli estremi di quelle consistenti
- Ad esempio spazio delle curve polinomiali, con obiettivo minimizzare: $min_w 1/n \sum_{i=1}^{n} (y_i - wx_i)^2$
- Training/empirical error vs ideal error
- Regularization con coefficiente di penalità alpha: $min_w 1/n \sum_{i=1}^{n} (y_i - wx_i)^2 + \alpha ||w||^2$
- Formula per calcolare w da X e y: $w = (X^T X + \alpha I)^{-1} X^T y$

PAC, generalization e SRM

- PAC (Probably Approximately Correct)

$$P(|\sigma - \pi| \geq \epsilon) \leq 2e^{-2\epsilon^2 N}$$

$$N \geq \frac{1}{\epsilon} ln(\frac{|H|}{\delta})$$

$$P(|\sigma - \pi| \geq \epsilon) \leq M2e^{-2\epsilon^2 N}$$

- In genere c'è molto overlapping di bad events: $m_H(N) \ll 2^N$
- Shattering $\forall S' \subseteq S, \exists h \in H, \forall x \in S,\ h(x) = 1\ sse\ x \in S'$
- VC-dimension: $max|S|_{S \subseteq X} : H\ shatters\ S$
    - La VC dimension dello spazio di iperpiani in R^n è $n + 1$
    - Fortunatamente in genere i punti della stessa classe sono in una stessa densità di distribuzione, quindi non serve preoccuparsi per tutte le possibili labellings dei punti

- Generalization error con VC-confidence $error(g) \leq error_S(g) + F(\frac{VC(H)}{n}, \delta)$
    - Inversamente proporzionale a n e δ, direttamente proporzionale ad VC(H)
    - SRM (Structural Risk Minimization) per il miglior trade-off tra A e B

Decision trees

- Struttura nodi interni e foglie
- Equivalenza con logica proposizionale in disjunctive normal form (DNF)
- Quando usarli
    - Classificazione
    - Facile interpretazione (motivi legali)
    - Veloce valutazione $log(depth)$
    - Non sono parametrici ma nemmeno conservano il dataset dopo il training
- CART e ID3
- Scelta dell'attributo ottimo (non sempre possibile, svantaggio del pre-pruning con XOR)
    - Entropia: $\sum_c^m p_c \; log \; p_c$ con $p_c = \frac{|S_c|}{|S|}$
    - Gini Index: $1 - \sum_c^m p_c^2$
    - Miclassification: $1 - max_c(p_c)$
- Information gain: $G(S, a) = E(s) - \sum_{v \in V(a)} \frac{|S_v|}{|S|} E(S_{a=v})$
- Bias del decision tree: l'information gain favorisce attribute con tanti valori discreti
    - Lo spazio delle ipotesi è l'insieme degli alberi
    - La tecnica è hill-climbing, quindi greedy
- Valori continue
    - Calcolo splitting point nel mezzo tra due istanze di classe diversa
    - L'attributo può essere riutilizzato
- Regressione: si calcola la media dei punti che raggiungono il nodo e si minimizza il MSE dalla media (varianza)
- Multivariate trees: permettono di definire boundary lineari ma non allineate agli assi, più flessibili
- Overfitting
    - Depth massima
    - Numero minimo di istanze in un branch, 5% del dataset
    - Entropia > θ
    - Pre-pruning

        - Rischia di sbagliare se manca attributo ottimale quando serve combinazione di attributi come in XOR
    - Post-pruning con errore validation set o ipotesi nulla con confidence $1 - \delta$
    - Rule post-pruning con regole ordinate per performance
- Istanze con valore mancante per attributo a
    - Valore più comune in X
    - Valore più comune in base a y
    - $n = |V(a)|$ rimpiazzi con peso $P(a = v|Tr)$ con $v \in V(a)$
    - Mean imputation
    - Imputation by regression

Neural networks

- Quando usarli
    - Sia classificazione che regressione
    - Alta dimensionalità
    - Accettabile tempo di learning alto ma tempo di valutazione basso della funzione appresa
    - Non è richiesta comprensibilità (black box)
    - Speech and image recognition
    - Adattamento con online learning
- Perceptron $y = sign(wx + b)$ oppure $y = sign(wx)$ con $w_0 = 1$
    - Calcola la discriminante lineare
    - Equivalenza con funzioni booleane AND, OR, NOT ma non con XOR
- base learning rule (perceptron)
    - Scelto $x \in S$ random
    - $w \leftarrow w + \eta(t - o)x_i$ con $o = sign(w \cdot x)$ e $t \in \{-1, +1\}$
        - Se la predizione è sbagliata $t = 1, o = 0$ e $x_i \geq 0$ allora il peso cresce per avvicinarsi a 1, altrimenti se $x_i \leq 0$ decresce
    - Se lo spazio è linearmente separabile, termina in numero di passi finito
    - w sono inizializzati random in un intorno di zero. Fermare le iterazioni equivale a fare regularization
    - Se η è troppo largo potrebbe causare troppe oscillazioni e perdersi il minimo, se è troppo piccolo la convergenza è troppo piccola
- activation function / threshold function
    - hard threshold (non derivabile): $sign(w \cdot x)$
    - sigmoide / logistic function (derivabile): $\frac{1}{1+e^{-w \cdot x}}$
- Hidden layers

    - Senza HL la decision surfaces ha un boundary lineare
    - Con HL le decision surfaces non sono lineari e le varie regioni sono unite dall'AND delle unità output
- Gradiente:
    - > Il gradiente di una funzione $f : R^n \rightarrow R$ è una funzione vettoriale $\nabla f : R^n \rightarrow R^n$. Spesso definito come il vettore che ha per componenti le derivate parziali della funzione.
    - L'algoritmo discesa di gradiente permette di trovare il vettore R^n che minimizza la funzione f, ovvero $x^* = argmin_x f(x)$
- delta rule:
    - attivazione lineare, senza hard-threshold: $o = w \cdot x$
        - $E[w] = \frac{1}{2N} \sum_i^n (t^i - o^i)^2$ da minimizzare rispetto a w
            1. Assegna a w_i valori random in $[-0.01, 0.01]$ vicini allo zero
            2. $\Delta w_i \leftarrow 0$
            3. $\forall (x, t) \in S$: $\Delta w_i \leftarrow \Delta w_i + \eta(t - wx_i)x_i$
            4. $\forall i \in \{1, ..., n\}$: $w_i \leftarrow w_i + \Delta w_i$
    - sigmoide: $o = \sigma(w \cdot x)$, $y = w \cdot x$
        - $\Delta w_i \leftarrow \Delta w_i + \eta(t - o)\sigma(y)(1 - \sigma(y))x_i$
- Multilayer:
    - d unità di ingresso: $x = (x_1, ..., x_d)$
    - N unità nascoste: $y = (y_1, ..., y_N)$
    - c unità di output: $z = (z_1, ..., z_c)$
    - w_{ji} peso da x_i a y_j
    - w_{kj} peso da y_j a z_k
- Backpropagation-1hl-stocastico:
    1. Calcola y e z con x
    2. $\forall z_k$
        - $\delta_k = z_k(1 - z_k)(t - z_k)$
        - $\Delta w_{kj} = \delta_k y_j$
    3. $\forall y_j$
        - $\delta_j = y_j(1 - y_j) \sum_{k=1}^c w_{kj}\delta_k$
        - $\Delta w_{ji} = \delta_j x_i$
    4. $w_{sq} \leftarrow w_{sq} + \eta \Delta w_{sq}$
- Teorema di universalità:
    - > Data una funzione continua $f : R^n \rightarrow R$, esiste un intero M tale che una rete neurale con almeno M unità nascoste sia in grado di calcolare una funzione $\hat{f} : R^n \rightarrow R$ e $max|f(x) - \hat{f}(x)| < \epsilon$ con qualunque $\epsilon > 0$.

- Difficoltà:
  - Topologia della rete e numero unità nascoste
  - Learning rate η
    - Adaptive learning rate $\Delta\eta = +a$ se l'errore diminuisce, $\Delta\eta = -b\eta$ altrimenti
  - Tempo di apprendimento
    - Aggiunta di un termine momento $\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1}$
  - Minimi locali
    - Apprendimento stocastico
    - Reti diverse con diverse inizializzazioni o ensemble
- Autoencoders
- Overfitting con troppe iterazioni o unità nascoste
  - Regularization sui pesi w
  - Approccio costruttivo o distruttivo
- Utilizzo di hints come immagine rotata e termine di errore per non riconoscimento degli hints

GLM (Generalized Linear Models) & SVM (Support Vector Machines)

- Margin: $\rho = 2r = \frac{2}{||w||}$
- Link with SRM: la VC-dimension degli iperpiani ottimali è $VC_{opt} \leq min\{\lceil \frac{R^2}{\rho^2} \rceil\} + 1$
  - Il true risk può essere minizzato massimizzando il margine $\rho = \frac{2}{||w||}$, quindi minimizzando $||w||$
- $min_{w,b} \frac{1}{2}||w||^2$ subject to $\forall i \in \{1, ..., n\} : y_i(wx_i + b) \geq 1$
  - Convex quadratic problem con linear constraints: può essere risolto analiticamente senza iterazioni come gradient descent
- Duale con coefficienti di Lagrange
- $max_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$ subject to $\forall i \in \{1, ..., n\}$ $\alpha_i \geq 0, \sum_i^n y_i \alpha_i = 0$
- Support vectors $x_i : \alpha_i \geq 0$
- $w = \sum_i^n y_i \alpha_i x_i$
- $b = y_k - wx_k$ per $\alpha_k \geq 0$
- $h(x) = sign(wx + b) = sign(\sum_i^n y_i \alpha_i (x_i x) + b)$
- Non linearmente separabile
  - $min_w \frac{1}{2}||w||^2 + C \sum_i^n \xi_i$ subject to $\forall i \in \{1, ..., n\}$ $\xi \geq 0, y_i(wx_i + b-) \geq 1 - \xi_i$

    - Hinge loss, simile a cross-entropy e più robusto di misclassification error o square error
- Kernel functions
    - Basis functions: $x \rightarrow \phi(x)$ con $\phi(x) = [\phi_1(x), ..., \phi_M(x)]$
    - $w = \sum_i^n y_i \alpha_i \phi(x_i)$
    - $h(x) = sign(\sum_i^n y_i \alpha_i (\phi(x_i) \cdot \phi(x)) + b)$
    - $K(x_k, x) = \phi(x_k) \cdot \phi(x) = K(x, x_k)$
    - $h(x) = sign(\sum_i^n y_i \alpha_i (K(x_i, x) + b)$
    - Kernel o gram matrix, simmetrica e positiva per definizione
    - Mercer's condition
- Possible kernels
    - Linear kernel, $K(x, x') = x \cdot x'$
    - Polynomial kernel: $K(x, x') = (x \cdot x' + u)^p$
    - Radial basis function: $K(x, x') = exp(-\gamma ||x - x'||^2)$
- Regression $min_{w,b,\xi,\xi^*} \frac{1}{2}||w||^2 + C \sum_i^n (\xi + \xi^*)$ subject to:
    - $\forall i \in \{1, ..., n\} \xi_i, \xi_i^* \geq 0,\ y_i - (w \cdot x_i + b) \leq \epsilon + \xi_i,\ (w \cdot x_i + b) - y_i \leq \epsilon + \xi_i^*$
- Kernel trick
    1. Qualsiasi algoritmo che possa essere riscritto come dot product dei suoi input, può sfruttare il kernel trick per miglior efficienza di computazione
    2. Utilizzo di algoritmi con input vettoriale per oggetto non vettoriali. La kernel function misura la similarità tra gli oggetti: $d(x, z) = ||\phi(x) - \phi(z)|| = \sqrt{K(x, x) + K(z, z) - 2K(x, z)}$
        - Stringhe (DNA), alberi, graphi
- Combinazione di kernel: somma, prodotto per costante, prodotto di kernels
    - Si possono anche usare dei pesi e diventa una sorta di ensemble di kernels, in modo da combinare l'informazione di similarità da diversi kernels
- Quando usarli
    - Input non vettoriali, grazie alle funzioni kernel
    - Utilizzo di kernel matrix invece del dataset originale, NLP e bioinformatica
    - Speech and image recognition
- One-class classification $min\ R^2 + C \sum_i \xi_i$
- Kernel PCA che usa PCA sulla kernel matrix, facendo una linear dimension reduction nello feature space

Preprocessing

- Feature categoriche/simboliche

    - Nominali vs Ordinali
    - OneHot Encoding con dummy variables
    - Bag of words
- features quantitative/numeriche
    - Intervalli vs Reali
    - $\hat{x}_j = \frac{1}{n}\sum_i^n x_{ij}$ e $\sigma_j = \sqrt{\frac{1}{n}\sum_i^n (x_{ij} - \hat{x}_j)^2}$
    - Centering: $c(x_{ij}) = x_{ij} - \hat{x}_j$
    - Standardizzazione: $s(x_{ij}) = \frac{c(x_{ij})}{\sigma_j}$
        - z-normalization
    - Scaling: $h(x_{ij}) = \frac{x_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}}$
    - Normalizzazione: $g(x) = \frac{x}{||x||}$
- Il k-NN richiede normalizzazione esempi, come pure k-means o la rete neurale
- Feature selection
    - Forward e backward selection
    - Non adatto per immagini
- Feature extraction
    - PCA: autovettori ed autovalori
    - LDA

Model selection e validation

- Bias $E[\hat{f}(x)] - E[f(x)]$ e varianza $E[(f(x) - E[f(x)])^2]$
- Underfitting/overfitting
- Rasoio di Occam
- Cross validation
- K-fold cross-validation
    - $k = |Tr|$ LOOCV
    - Bias/variance col cambiare di k
    - Per selezione degli iper-parametri
- Accuracy, non ideale se ci sono tanti esempi positivi rispetto ai negativi
- Contingency table
    - Precision: $\pi = \frac{TP}{TP+FP}$ (degree of soundness)
    - Recall: $\rho = \frac{TP}{TP+FN}$ (degree of completeness)
- F-measure $F_\beta = \frac{(1+\beta^2)\pi\rho}{\beta^2\pi+\rho}$
    - Con $\beta = 1$ si ha $F_\beta = \frac{2\pi\rho}{\pi+\rho}$
- Multiclass classification
    - One-vs-all

    - One-vs-one (pairwise)
    - Confusion matrix: precision in colonna, recall in riga
    - Micro/macro-averaging

Representation learning

- Miglior rappresentazione degli input per migliorare la regressione o classificazione
- PCA
    - Utilizza la matrice S, Σ stimata sul dataset
    - Conviene standardizzare il dataset nel pre-processing
    - Scree graph
    - Plottare le due principal components
    - Minimum reconstruction error
        - Usato per lossy compression
    - Si può usare il kernel trick per mappare da X a $\phi(X)$, ottenendo una non-linear reduction delle dimensioni originali
- Autoencoders
    - Unsupervised con NN
    - Bottleneck hidden layer
    - Si cerca di minimizzare la funzione reconstruction error $||x - \tilde{x}||_2^2 con $\tilde{x} = g_{w'}(f_w(x))$
    - L'apprendimento viene fatto con SGD
    - Deep non linear autoencoders con più hidden layers, progettano in piani non lineari
    - Regularized autoencoders per favorire anche sparsity della rappresentazione oppure robustezza al rumore
        - Sparsity significa che nella rappresentazione molti valori sono a zero
- Word embedding trasforma dal testo a vettore di parole
    - Word2vec cerca di mantenere la semantica e il contestoreconstruction error
- Knowledge graph, nodi come entità e archi come relazioni
    - Rappresentazione concisa delle relazioni tra le entità

Apprendimento Bayesiano

- $P(h|D) = \frac{P(D|h)*P(h)}{P(D)}$
- $h_{MAP} = argmax_{h \in H} P(D|h) = h_{ML}$

- Nel caso dell'apprendimento di una funzione reale con target a distribuzione normale
  - $p(d_i|h) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(d_i-h)^2}$
  - $h_{ML} = argmax_{h\in H}\ p(D|h) = argmax_{h\in H}\ \prod p(d_i|h) = argmax_{h\in H}\ \sum ln(p(d_i|h))$
- Nel caso di una rete neurale in cui si vuole la probabilità della classificazione invece che 0/1 si vede che la h_{ML} minimizza la cross-entropy, migliore dello square error.
- Classificazione ottimale di Bayes, pesata dalle probabilità a posteriori $P(v_j|D) = \sum_{h\in H} P(h|D)P(v_j|h)$
- Classificatore di Gibbs $E[\epsilon_{Gibbs}] \leq 2E[\epsilon_{Bayes}]$
- Naive Bayes
  - Quando usarlo
    - Dataset grandi
    - Classificazione di documenti (newsgroups con accuracy 86%)
  - $v_{MAP} = argmax_{v\in V} P(a_1, ..., a_m|v)P(v) = argmax_{v\in V} P(v)\prod_i^m P(a_i|v)$
  - L'assunzione naive di indipendenza condizionale è spesso violata, ma l'algoritmo funziona comunque senza stimare correttamente le probabilità a posteriori, a patto che $argmax_{v\in V} \hat{P}(v)\prod_i^m \hat{P}(a_i|v) = argmax_{v\in V} P(v)\prod_i^m P(a_i|v)$
  - La probabilità a posteriori del NV tende a 1 o 0 essendo produttoria
  - m-stima di probabilità con valori virtuali $\hat{P}(a_1|v_j) = \frac{n_c+mp}{n+m}$
    - m è chiamato equivalent sample size
- Expectation Maximization (EM)
  - Sceglie ipotesi iniziale random $h = \langle \mu_1, \mu_2 \rangle$
  - Passo E: calcola il valore atteso $E[z_{ij}]$ assumendo valga l'ipotesi $E[z_{ij} = \frac{p(x=x_i|\mu=\mu_j)}{\sum_j^2 p(x=x_i|\mu=\mu_j)}]$
  - Passo M: calcola la nuova ipotesi h_{ML}, assumendo i valori attesi $E[z_{ij}]$: $\mu_j = \frac{\sum_i E[z_{ij}]x_i}{\sum_i E[z_{ij}]}$

Ensemble learning

- Vogliamo learners con alta accuracy ma il più diversi possibili
- Justification:
  - Statistical: by "averaging" the votes of several "good" classifiers the risk of choosing the wrong classifier is reduced

    - Computational: an ensemble constructed by running the local search from many different starting points may provide a better approximation to the true unknown function, avoiding to be stuck in local minima
    - Representation: forming weighted sums of hypotheses drawn from H it may be possible to expand the space of representable functions
- Generalization error $E[(y - g(x))^2]$ = noise^2 + bias^2 + variance
- Parallel
    - Voting
        - $P(H(x) \neq f(x)) \leq e^{-T/2(2\epsilon-1)^2}$
        - Purtroppo in realtà gli errori dei votanti non è indipendente
    - Bagging
        - Bootstraping
        - The prediction is $H(x) = 1/k \sum_i^k h_i(x)$
        - In teoria il bias rimane invariato e la varianza diminuisce, nella pratica il bias aumenta perché i weak learners non sono indipendenti
        - Funziona bene con unstable learners, con alta varianza, come decision stumps
    - Random forests
        1. Usa bootstrap
        2. Utilizza features random come nodi interni
        3. Aggrega le predizioni
- Sequential/Boosting
    - Usa weak-learners con $accuracy = 0.5 + \epsilon$
    - AdaBoost: $H(x) = sign(\sum_t \alpha_t h_t(x))$
        - Inizialmente $p_t^i = 1/N$
        - Ad ogni passo, l'errore è ϵ e $\beta = \frac{1-\epsilon_t}{\epsilon_t}$ e $p_{t+1}^i = \beta p_t^i$ se l'istanza x_i è stata classificata correttamente, altrimenti rimane p_t^i
        - I pesi dei weak learners sono $w_t = log(\frac{1}{\beta_t})$
        - Adaboost non fa overfitting e riduce il bias dei weak-learners, che però hanno bassa varianza
        - Sensibile al rumore
- Stacking
    - Diversi tipi di learners
    - Combinazione non lineare dei learners per aggiustare il loro bias
    - Richiede apprendimento dei parametri della combinazione non lineare su un diverso dataset
    - Learners complementari con diversi bias induttive
- Gli ensemble learning sono usati in NLP con labbiale oppure in sistemi biometrics per autenticazione da diversi input

Clustering

- Criteri interni di valutazione
  - Similarità intra-class
  - Similarità inter-class
  - Misura di similarità, ad esempio norma euclidea
    - Si può usare anche la distanza di Minkowski in generale o la matrice di Mahalonobis
- Criteri esterni di valutazione
  - Classificazione esterna ground-truth
  - Purity: $\frac{|c|}{|K|}$ con c classe dominante e K il cluster
  - RandIndex: simile alla contingency table, considerando coppie di esempi
    - Si possono considerare le equivalenti della precision e della recall
- Algoritmi di partizionamento
  - k-means
    - Centroide $\mu(c) = \frac{1}{|C|} \sum_{x \in C} x$
    - Ad ogni esecuzione si riduce il valore della funzione obiettivo $V(D, \gamma) = \sum_k^K \sum_{i:\gamma(d_i)=k} ||d_i - c_k||^2$
    - Usato per quantizzazione dei colori
    - Possibile per dimension reduction
- Algoritmi gerarchici
  - Bottom-up HAC (Hierarchycal Agglomerative Clustering)
    - Single-link: minor similarità di una coppia $O(N^2)$
    - Complete-link: massima similarità di una coppia, $O(N^2 log(N))$
    - Average-link: media similarità di tutte le coppie, $O(N^2 log(N))$
    - Centroid: distanza dei centroidi, $O(N^2 log(N))$
  - Top-down
  - Dendrogramma
- Scelta del numero di clusters
  - Plottare i dati sulle componenti PCA per vedere i clusters
  - Controllare la qualità dei clusters nel dominio, ad esempio visual check in color quantization

Recommender systems

- Content-based: similarità degli elementi con quelli che l'utente ha già visto
- Context-aware

- Collaborative filtering: similarità delle interazioni
  - Similarità item-item o utente-utente
  - Feedback espliciti $r_{ui} \in [1, 5]$ ed impliciti $r_{ui} \in 0, 1$
  - Usano interaction matrix, o nel migliore dei casi rating matrix
  - La task è apprendere l'utilità di ogni item per ogni utente
  - Matrice molto grande, sparsa e con valori mancanti
    - Density < 0.01%
    - User activity e items popularity hanno caratteristica distribuzione con lunga coda
- Approccio ibrido
  - CB > CF senza history
  - CF > CB con tante interazioni rispetto a reviews esplicite
- Recommendation tasks
  - Predire il rating
  - Suggerire i top-N non visti che piacerebbero di più
- Indici di qualità
  - Rilevanza: piacciono
  - Coverage: quanti di quelli che piacciono
  - Novelty $\frac{\sum_{i,j} 1-sim(i.j)}{m(m-1)}$
  - Diversità: $\frac{\sum_i log_2(\frac{1}{popularity(i)})}{|TP|}$ oppure $\frac{\#rilevanti\ e\ unknown}{\#rilevanti}$
  - Serendipity: sorprendere l'utente, non avrebbe trovato da solo
- Errori di predizione nel test set Te
  - MAE: $\frac{1}{|Te|}||r_{ui} - \hat{r}_{ui}||^1$
  - MSE: $\frac{1}{|Te|}||r_{ui} - \hat{r}_{ui}||^2$
  - RMSE: $\sqrt{\frac{1}{|Te|}||r_{ui} - \hat{r}_{ui}||^2}$
- top-N
  - Recall e precision
  - AUC
  - AP (Average Precision)
  - DCG (Discounted Cumulative Gain)
  - MRR (Mean Reciprocal Rank)
- Non-personalized RS
  - Most popular
  - Highest rated: con normalizzazione per favorire popolarità
- Algoritmi di recommendation
- k-NN: trova i k utenti più simili all'utente a e suggerisce i film non visti da a
  - $sim(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||}$
  - $sim(i, j) = \frac{i \cdot j}{||i|| \cdot ||j||}$

    - Pearson correlation
- Matrix factorization: cerca di ridurre la dimensione della matrice da $n \times v$ in 2+ matrici $n \times k$ e $k \times v$ di componenti latenti P e Q
    - p_u misura l'interesse dell'utente per i fattori
    - q_i misura quanto l'item possiede i fattori
    - $L = ||R - P \times Q^T||^2$
    - $\hat{r}_{ui} = p_u^T q_i$
    - $argmin_{q,p} \sum_{(u,i) \in Tr} (\hat{r}_{ij} - p_i^T q_j)^2 + \lambda(||p_u||^2 + ||q_i||^2)$
    - Discesa di gradient stocastica per trovare il minimo della derivata
    - ALS (Alternate Least Square): A turno, fissando una delle matrici latenti, ad esempio P, il problema di ottimizzare l'altra, Q, diventa convesso con soluzione analitica
        - Procedura ripetuta per diverse iterazioni