

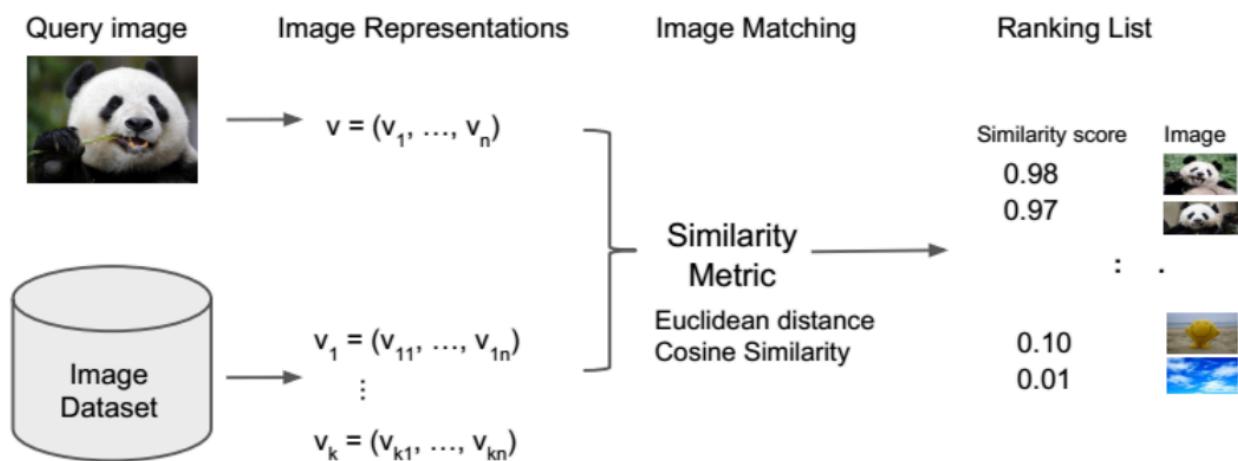
Lezione 8 - Content Based Image Retrieval - 25/11/2024

Content Based Image Retrieval

Abbiamo un'immagine di query, e un database di immagini, vogliamo ritornare una lista dove la prima è quella più "related", etc.

Per confrontare l'immagine di query con quelle nel dataset, dobbiamo calcolare una signature che descriva ciascuna immagine in modo compatto.

Quindi creiamo una rappresentazione (di stesso tipo) per ciascuna immagine all'interno del dataset, e per quella di query. Poi abbiamo bisogno di verificare quali immagini sono vicine a quella di query. Dobbiamo definire una metrica di similarità. L'output sarà ordinato in base al similarity score, quella più relazionata sarà la prima (di solito la similarità è in un range 0-1. Di solito vengono mostrate all'utente le top X immagini, non tutte).



In base a che tipo di similarità ci interessa, alcune features possono essere più utili di altre. Il feature vector potrebbe anche essere estratto da un modello NN pre-trainato.

Potrei essere più interessato al contenuto, oppure al colore, alle textures... dipende dalla task.

Key problems

Ci sono dei problemi chiave:

- Che tipo di databases? (image domain)
- Che tipo di queries vogliamo che l'utente possa fare? (query domain)
- Cos'è un match? Che similarità usiamo? (image similarity)
- Come possiamo valutare la performance a livello numerico?
- Come possiamo rendere tutte le ricerche delle query efficienti?

Image domain

Il **narrow domain** significa che abbiamo una variabilità limitata e predicibile di tutti gli aspetti di un'immagine. Per esempio abbiamo immagini di tv news di un canale specifico. Quindi le persone saranno sempre le stesse, alcune cose saranno sempre uguali, ma per esempio c'è variabilità nei vestiti, ... in alcune immagini potremmo avere più di un presentatore... non abbiamo molta variazione.

- ▶ Narrow domain
 - ▶ Limited and predictable variability of relevant aspects of image appearance



D'altro canto, abbiamo il **broad domain**, dove abbiamo una variabilità senza limite e non predicibile di aspetti relativi all'apparenza delle immagini. Per esempio negli sport, potremmo avere sport diversi, potrebbero esserci immagini molto recenti con immagini molto vecchie grayscale e di bassa qualità, ci può essere una variazione alta su come le immagini sono catturate (da vicino, da lontano...)...

► Broad domains

- Unlimited and unpredictable variability of relevant aspects of image appearance



Query domain

Qual è il tipo di queries che vogliamo poter fare nel nostro sistema?

Nel contesto di **ricerca del target o di un'istanza** possiamo avere queste query per esempio:

► Target search / instance search

- Does a given image exist as is in the database?
- Does a scaled and/or rotated version of this image exist in the database?
- Does a region of this image exist in the database?
- Does an object exist in the database?



Queste sono tutte target search o instance search (controlla se esiste quell'esatto oggetto nel db).

Nel contesto di **ricerca di categorie** abbiamo:

- ▶ Category search
 - ▶ Retrieve images belonging to the same category
 - ▶ E.g. retrieve all images of «chairs»



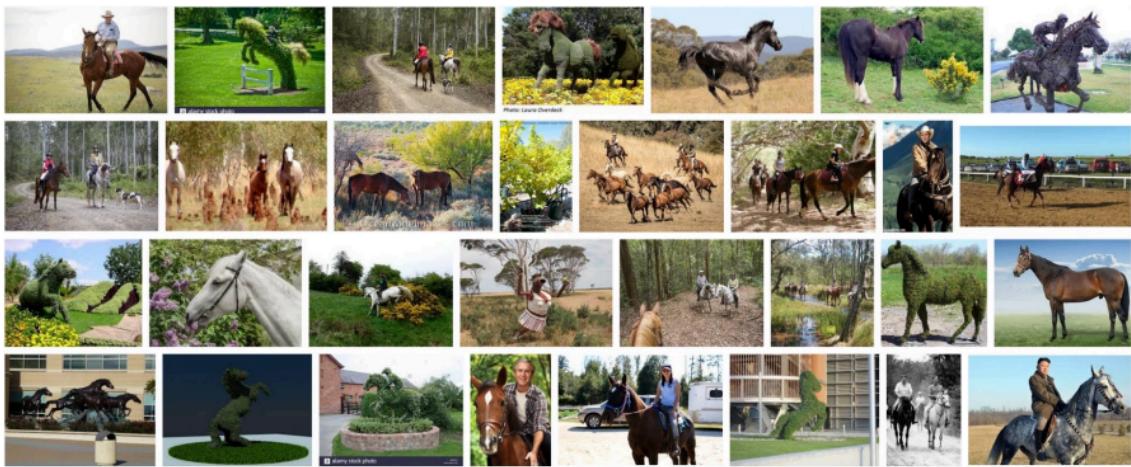
Qui il primo risultato sarà quello più simile (per colore, forma, posizione...) e man mano quelle meno simili.

Nel contesto della **search by association / similarity search** abbiamo:

- ▶ Search by association / similarity search
 - ▶ Retrieve all the images that are (overall) similar to the query



Come possiamo descrivere quello di cui l'utente ha bisogno? Usiamo keywords e/o descrizioni? Per esempio "trova un'immagine con un cavallo e un cespuglio".

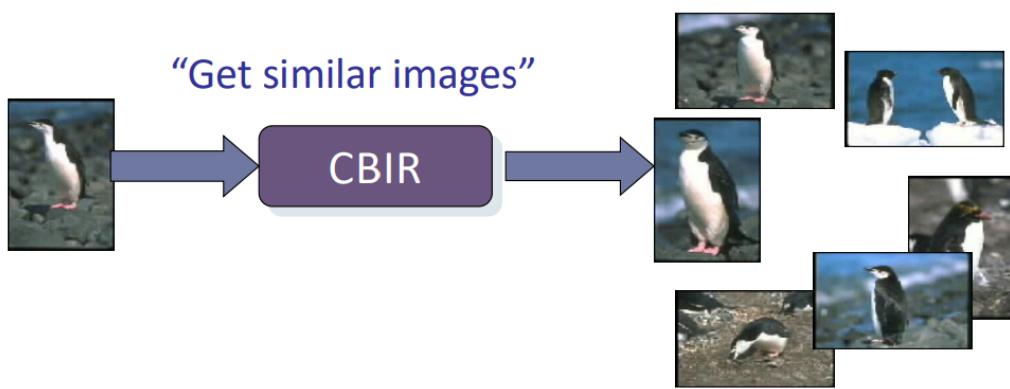


Query by example

Uno dei modi più comuni per formulare la query è la **query by example**, dove diamo al sistema uno o più esempi che siano pertinenti alla query (tra quelle del database o al di fuori del db).

► Query by example (QBE)

- Provide to the system one or more example relevant to the query (within or outside the database)



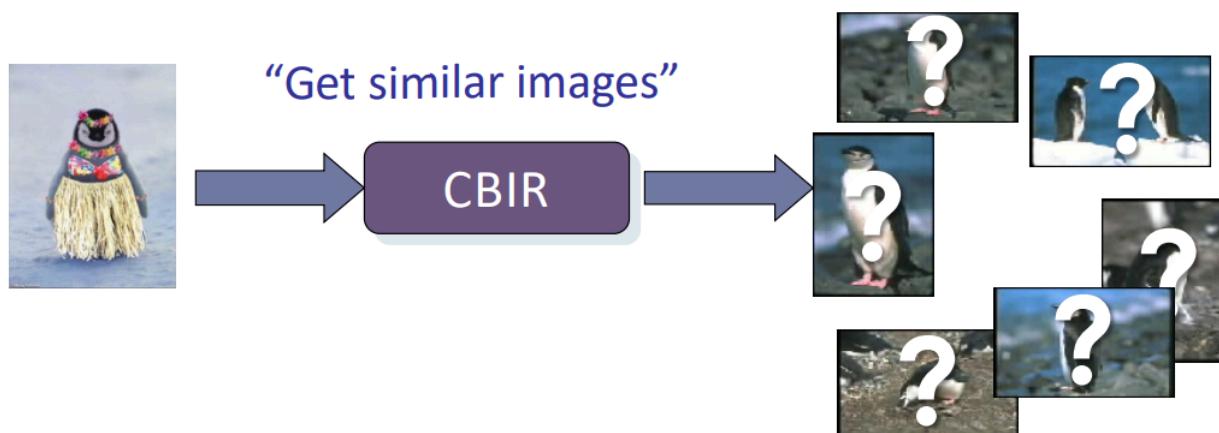
Facendo questo abbiamo una descrizione indiretta del contenuto della query che è più flessibile del testo (non serve una traduzione in lingue diverse, per esempio).

è più semplice per un'utente mostrare un esempio di cosa sta cercando, piuttosto che scrivere una descrizione di quello che vuole.

La query in questo modo diventa più compatta, naturale e conveniente per l'utente.

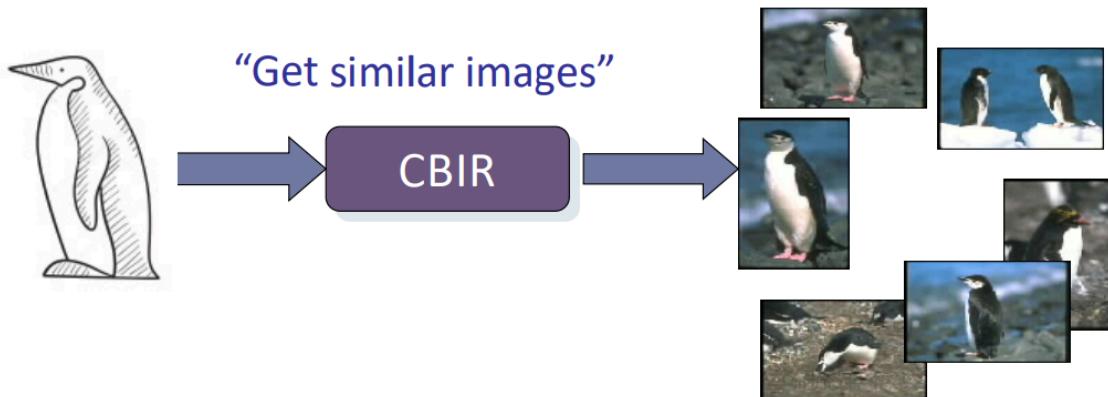
Il risultato può essere valutato facilmente dall'utente, facendo una comparazione del risultato rispetto alla query.

Il problema di questo metodo è che bisogna fornire buoni esempi.



Query by sketch

Potrebbe anche succedere che non ho l'immagine originale. Ci sono dei tipi di query che si chiamano **query by sketch**, dove si disegna una descrizione visiva di come dovrà essere l'immagine.



Query by multiple examples

Potremmo anche voler esprimere dei concetti che non possono essere descritti da una sola immagine, questo è il caso di **query by multiple examples**, per esempio se voglio ritornare immagini di frutta, perché se usassi una sola immagine, probabilmente riceverei immagini di quel singolo frutto.

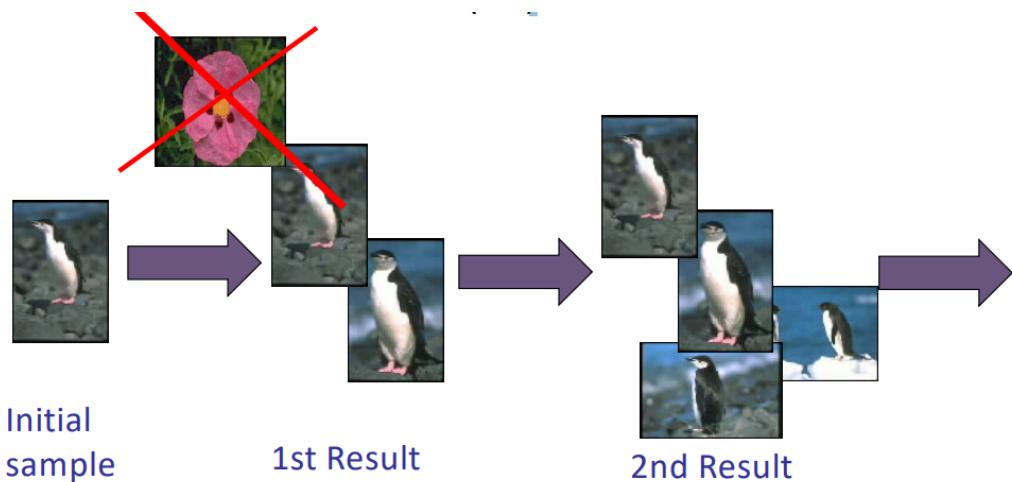
Relevance feedback (RF)

Il relevance feedback è anche chiamato "human in the loop", significa che l'utente è una parte attiva nella ricerca.

A causa di scelte soggettive e dell'incompletezza degli esempi, il CBIR potrebbe non restituire i dati corretti al primissimo passaggio di interrogazione.

Per esempio se abbiamo l'immagine di un pinguino che guarda a sinistra, il sistema non sa se vogliamo pinguini che guardino esattamente in quella direzione, non sa quali immagini siano rilevanti, rispetto a quello che l'utente vuole.

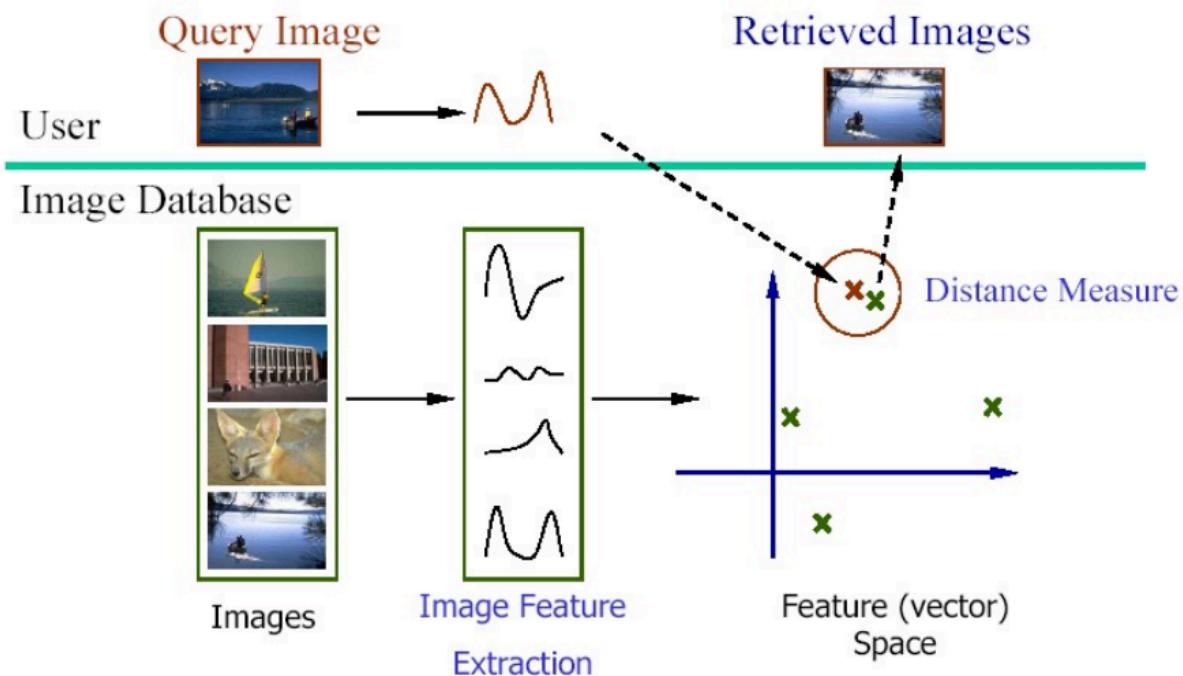
L'idea è che l'utente attivamente e iterativamente dia del feedback positivo e negativo al sistema.



L'utente potrebbe dire dare un feedback negativo sulla prima immagine per esempio.

In generale, questo è il tipo di operazione che è fatta. Da una parte abbiamo l'utente e dall'altra il database, abbiamo le immagini del db dove viene fatto feature extraction, facciamo la stessa cosa per la query image, poi tutte queste features

sono viste come punti in un feature space ad alte dimensioni, e quindi possiamo prendere le immagini che sono vicine nello spazio.



Abbiamo modi diversi di ritornare la nostra lista. Per esempio, siamo sicuri di voler ritornare tutte le immagini? Potremmo dire che dopo una certa soglia ci fermiamo.

Nel **range search** vogliamo trovare tutte le immagini con una distanza/score di similarità che è in un certo range $[s_{min}, s_{max}]$. Non restringe la grandezza della lista, ma potrebbe ritornare un set vuoto.

Poi abbiamo il **K nearest neighbour search**, che trova le k immagini che hanno la miglior distanza/score di similarità. Quindi la lista ritornata sarà al massimo lunga k. Le immagini recuperate indipendentemente dai loro effettivi punteggi di somiglianza/distanza rispetto alla query (cioè, la somiglianza può essere piuttosto bassa se non ce ne sono di migliori).

Abbiamo anche il **within distance search** (o alpha cut) che trova tutte le immagini che hanno un similarity score che è più alto di una soglia. Quindi non c'è limite alla lunghezza della lista ritornata, ma questa potrebbe essere vuota.

In ogni caso, abbiamo una **ranked list**, e mostreremo all'utente prima quella più simile.

Performance evaluation

Per poter avere una valutazione numerica del sistema, dobbiamo definire una collezione di immagini di test che sia statisticamente significativa al dominio.

Poi, per ciascuna test query, dobbiamo ottenere il relevance judgement (ovvero la ground truth), che si ottiene facendo pooling delle risposta di multipli utenti (questionario praticamente).

Possiamo avere diversi tipo di relevance judgement, si può avere positivo, rilevante/non rilevante, ranking...

In particolare, data una query, come possiamo confrontare i risultati del sistema rispetto alle ground truth che abbiamo?

- ▶ Precision

- ▶ «Ratio of the number of relevant documents retrieved over the total number of document retrieved»

- ▶ Recall

- ▶ «Ratio of relevant documents retrieved over the number of relevant documents in the collection»

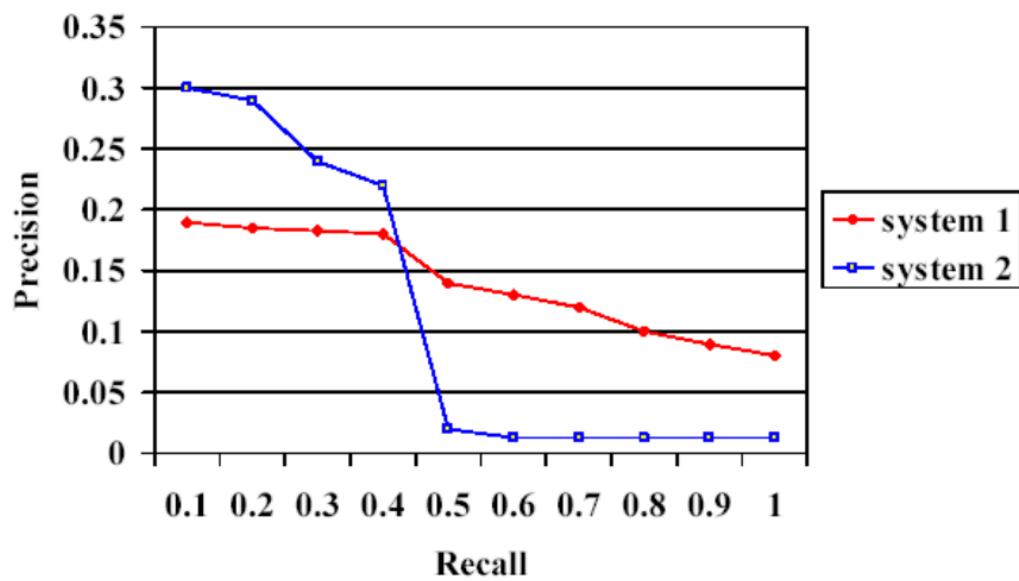
In generale, possono essere calcolate in questo modo:

► Standard Information Retrieval Evaluation Measures

- For a given query q :
- R_a = set of relevant documents in answer set A
- R = set of relevant documents for q

$$\text{Recall} = \frac{|R_a|}{|R|} \quad \text{Precision} = \frac{|R_a|}{|A|}$$

In questo modo possiamo usare entrambe, comparandole. In base alla task, l'utente potrebbe preferire una buona recall, una buona precision, oppure un compromesso.



Queste due metriche possono essere viste insieme con l'F-score:

- ▶ F-Score

- ▶ Accuracy of the retrieval system

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Ci sono altri metodi che sono più specifici per questo tipo di task con le immagini, per esempio questa:

(dobbiamo giusto sapere che esiste)

- ▶ Average Normalized Modified Retrieval Rank (ANMRR)
 - ▶ Proposed in the MPEG-7 standard

$$ANMRR = \sum_{q=1}^{NQ} \frac{AVR(q) - 0.5[1 + N(q)]}{1.25K - 0.5[1 + N(q)]}$$

- ▶ AVR(q) = average rank of relevant documents
- ▶ N(q) = number of relevant documents
- ▶ K = rank threshold

è importante valutare il sistema anche in termini di usabilità. Visto che dobbiamo matchare la feature della query rispetto a quelle del database, questo deve avvenire velocemente.

► Query Speed (Jackob Nielsen, 1993)

- ▶ 10s : maximum to keep user's attention (keep the user engaged)
- ▶ 1s : maximum for interactive working
- ▶ 0.1s : instantaneous response

(il secondo caso con 1s è quello del relevance feedback)

(il terzo caso con 0.1s è il massimo per considerare la risposta come istantanea)

D'altro canto, per definire l'usabilità del sistema, c'è questa system usability scale, dove abbiamo 10 domande a cui si può rispondere con un numero da 0 (strong disagree) a 4 (strong agree) e le risposte sono sommate, perchè ci sono alcune domande dove è meglio avere 0 e altre 4. Se lo score è di 68 o meglio allora il sistema è considerato meglio della media. Questo sistema è usato in molti campi diversi per l'usabilità del sistema (anche con domande diverse).

► System Usability Scale (John Brooke, 1986)

- ▶ Strong disagree (0) → strong agree (4)
- ▶ Scores are adjusted and summed up
- ▶ A score of 68 is above average

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Comparing feature vectors

Come possiamo computare il feature vector?

La comparazione è fatta in termini di distanza o similarità, in particolare:

La funzione d prende due immagini, l'output è un numero reale. Nel confronto tra due immagini abbiamo le proprietà di non negatività, simmetria, identità e triangle inequality.

- ▶ Comparison performed in terms of distance (or similarity)
 - ▶ Given a data domain \mathbb{D}
 - ▶ A metric is a function : $d: \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{R}$

non negativity	$\forall x, y \in \mathbb{D}, d(x, y) \geq 0$
symmetry	$\forall x, y \in \mathbb{D}, d(x, y) = d(y, x)$
identity	$\forall x, y \in \mathbb{D}, x = y \leftrightarrow d(x, y) = 0$
triangle inequality	$\forall x, y, z \in \mathbb{D}, d(x, z) \leq d(x, y) + d(y, z)$

Abbiamo molti modi diversi con cui possiamo comparare i feature vectors:

- If $\mathbb{D} = \mathbb{R}^n$ (i.e. vector space)

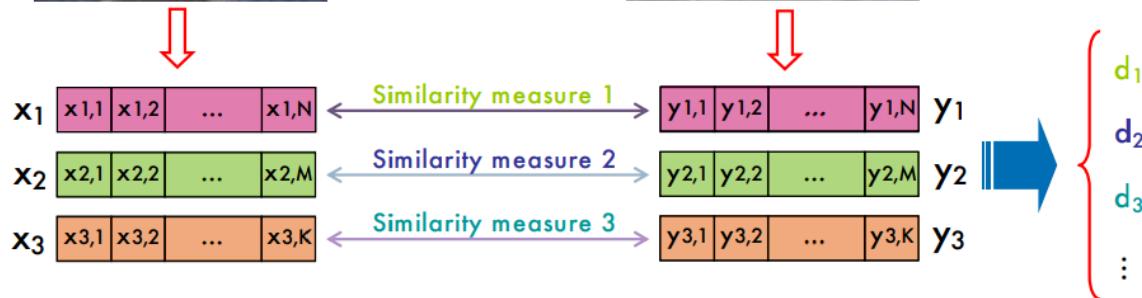
Euclidean	$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^n (x_i - y_i)^2 \right]^{1/2}$
Chebyshev	$d(\mathbf{x}, \mathbf{y}) = \max_{i=1..n} x_i - y_i $
Manhattan (city-block)	$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n x_i - y_i $
Minkowsky	$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^n x_i - y_i ^p \right]^{1/p}$
Weighted Minkowsky	$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^n w_i x_i - y_i ^p \right]^{1/p}$
Mahalanobis	$d(\mathbf{x}, \mathbf{y}) = \det(\mathbf{C}) (\mathbf{x} - \mathbf{y}) \mathbf{C}^{-1} (\mathbf{x} - \mathbf{y})$
Generalised Euclidean (Quadratic)	$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y}) \mathbf{K} (\mathbf{x} - \mathbf{y})$

una molto usata è la weighted Minkowsky

Fin ora abbiamo solo considerato una feature per l'intera immagine, ma in realtà potremmo avere diverse features che descrivono diversi aspetti dell'immagine.

- Feature space

- Set of all possible feature vectors



Quindi calcoleremmo diverse distanze, una per ogni feature.

Questo è fatto perchè avere più features funziona meglio.

- ▶ Using multiple features usually works better than single

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n] \\ \mathbf{Y} &= [\mathbf{y}_1, \dots, \mathbf{y}_i, \dots, \mathbf{y}_n] \end{aligned} \quad d(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^n \omega_i d_i(\mathbf{x}_i, \mathbf{y}_i)$$

Quindi ciascuna distanza avrà un peso diverso.

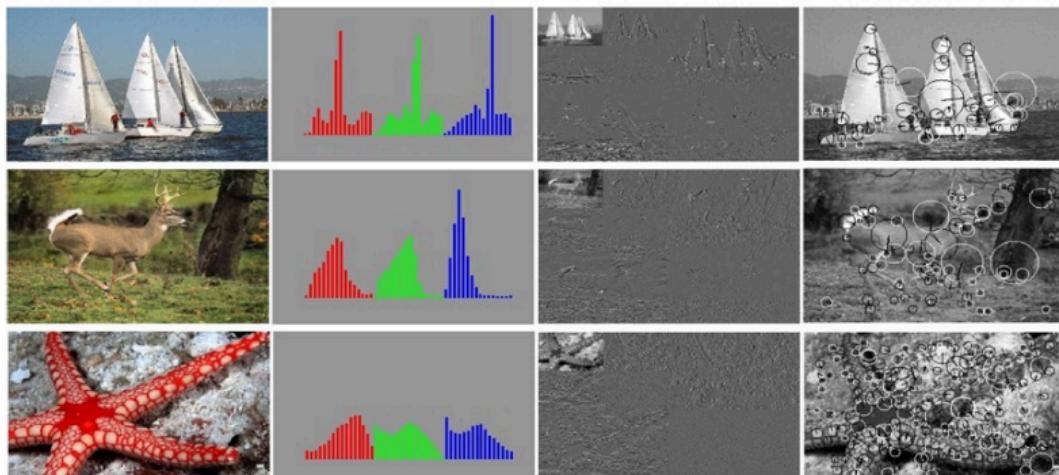
- ▶ Open issues:
 - ▶ Index may be very large: Efficiency
 - ▶ Choice of the best distance for each feature
 - ▶ The distances may be defined on different intervals
 - ▶ The weight should be set appropriately

L'idea generale è che prima di combinarle, dobbiamo normalizzarle, di modo che abbiamo valori nello stesso range.

Quali sono le proprietà ideali per le features dell'immagini per il CBIR?

- ▶ Ideal properties of image features for CBIR:
 - ▶ **perceptual similarity**: the feature distance between two images is large only if the images are not "similar";
 - ▶ **efficiency**: they can be rapidly computed;
 - ▶ **economy**: their dimensions should be small in order not to affect retrieval efficiency;
 - ▶ **scalability**: the performance of the system is not influenced by the size of the database;
 - ▶ **robustness**: changes in the imaging conditions of the database images do not affect retrieval.

Possiamo avere delle features low-level come l'istogramma, oppure high-level features che arrivano da diversi layers di neural network



- ▶ Low-level, hand-crafted features
- ▶ High-level, learned features

L'uso delle NN può funzionare se si hanno GPU, altrimenti è troppo lento.

I diversi layers dell'architettura danno informazioni diverse. Quelli più vicini all'output hanno distrutto l'informazione spaziale, e sono più correlati al contenuto

dell'immagine (utile per la similarità). Invece quelli convoluzionali iniziali hanno ancora informazioni spaziali, e quindi vedono la composizione.

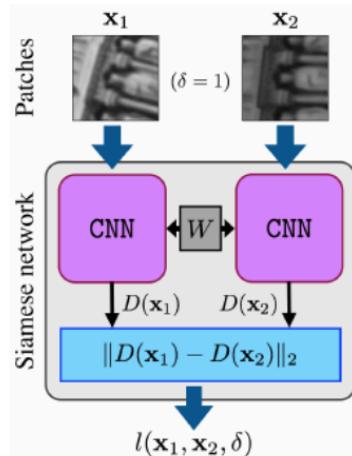
Fin ora abbiamo visto features che arrivano da un network che era fatto per un'altra task (per esempio classification di IMAGENET).

Potremmo usare un **siamese network**. Significa che la rete prende 2 diversi input, inviata a 2 diversi network che sono identici (copiati). Calcoliamo l'attivazione e la distanza tra i due risultati. In base al fatto che le 2 immagini iniziali sono related o no (label 1 o 0) facciamo il learning.

Così stiamo imparando una rappresentazione che è fatta apposta per la nostra task, il network non arriva da una task diversa. Questo metodo funziona meglio (bisogna avere abbastanza dati).

► Siamese Network

- Network to learn a function that **maps input patterns into a target space** such that ℓ_2 -norm in the target space approximates the **semantic distance** in the input space.
- Often used in face verification

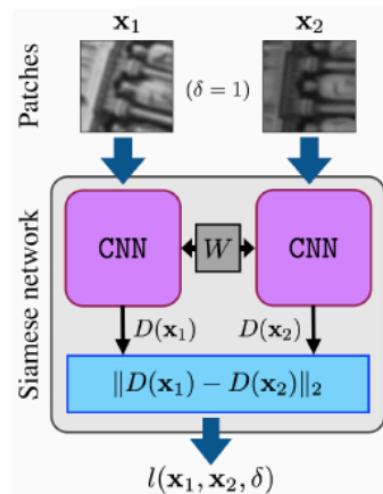


$$l(\mathbf{x}_1, \mathbf{x}_2, \delta) = \delta \cdot l_P(d_D(\mathbf{x}_1, \mathbf{x}_2)) + (1 - \delta) \cdot l_N(d_D(\mathbf{x}_1, \mathbf{x}_2))$$

$$\delta = \begin{cases} 1 & \text{if similar pair} \\ 0 & \text{otherwise} \end{cases}$$

$$l_P(d_D(\mathbf{x}_1, \mathbf{x}_2)) = d_D(\mathbf{x}_1, \mathbf{x}_2)$$

$$l_N(d_D(\mathbf{x}_1, \mathbf{x}_2)) = \max(0, m - d_D(\mathbf{x}_1, \mathbf{x}_2))$$

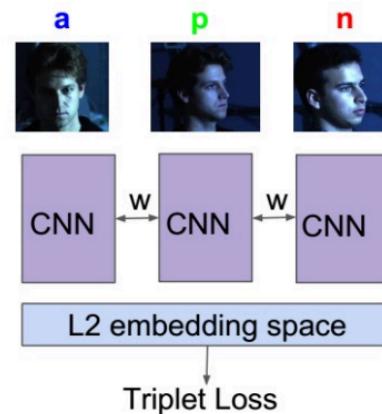
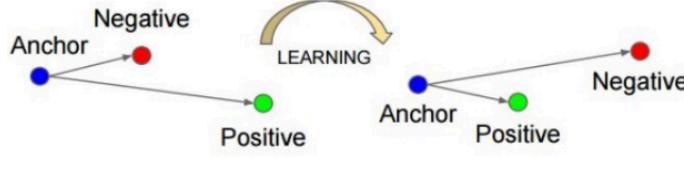


Un'estensione è la **triplet loss**, dove abbiamo 3 immagini di input, il primo esempio è l'anchor, poi abbiamo un esempio positivo (immagine in relazione alla prima) e un esempio negativo (immagine non in relazione alla prima).

Quindi impariamo i pesi di modo che i positivi siano più vicini all'ancora rispetto ai negativi.

► Siamese Network with Triplet Loss

- Loss function minimizes distance between query and positive and maximizes distance between query and negative



$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

Questo meccanismo che avvicina gli esempi in relazione e allontana quelli in non relazione probabilmente accade anche nel siamese network normale.

Search speed-up

Come possiamo velocizzare la fase di ricerca?

Abbiamo 2 diverse direzioni che possiamo seguire: trovare una soluzione esatta o una soluzione approssimata.

Il modo più semplice di avere la soluzione ottimale è la ricerca sequenziale, dove compariamo una per una l'immagine di query con tutte le immagini del dataset, però questo richiede molto tempo.

Sarebbe utile se potessimo escludere parte di questa lista. Supponiamo di aver creato una struttura dati di modo che se provo la distanza rispetto a 2 punti e poi rispetto ad un terzo che è più lontano, e la distanza è più alta, allora so che posso skippare altri punti che sono nella zona di quest'ultimo punto.

Più grande è il feature vector, più tempo sarà richiesto per la ricerca. Un modo per velocizzare la ricerca è di fare semplicemente feature reduction dei dati (per esempio PCA, prendendo il 95% della varianza).

Altrimenti, si può creare una data structure che renda i dati più facilmente ricercabili, come un tree.

Altrimenti si può fare data filtering: può succedere che l'immagine sia un pinguino, e il dataset sia di animali, quindi escludiamo tutti gli animali che non sono un pinguino.

(non abbiamo visto tutte le slides, ci sono degli extra)