Few-shot learning, Metric learning

Andrey Stotskiy & Vlad Shakhuro

30 October 2025

Outline

1. Introduction
    1.1. Domains and datasets
    1.2. Evaluation and metrics

2. Metric learning methods
    2.1. Sample-based methods
    2.2. Proxy-based methods

3. Efficient searching

Outline

1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

3. Efficient searching

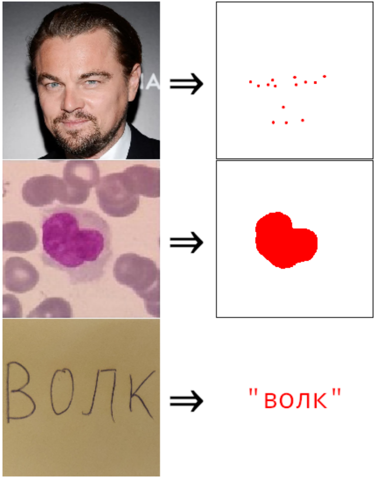
Can every task be reduced to classification?

Can every task be reduced to classification?

In theory – yes. In practice – no. Why not?

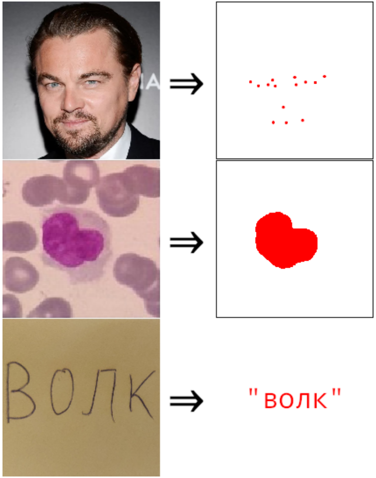
Can every task be reduced to classification?

In theory – yes. In practice – no. Why not?



Can every task be reduced to classification?

In theory – yes. In practice – no. Why not?



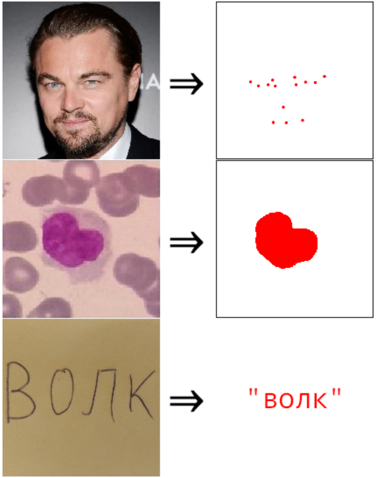
Is this Leonardo DiCaprio?

Task: Celebrity Actor Recognition

Can **this** task be reduced to simple classification?

Can every task be reduced to classification?

In theory – yes. In practice – no. Why not?



Task: Celebrity Actor Recognition

Can **this** task be reduced to simple classification?

Why not?

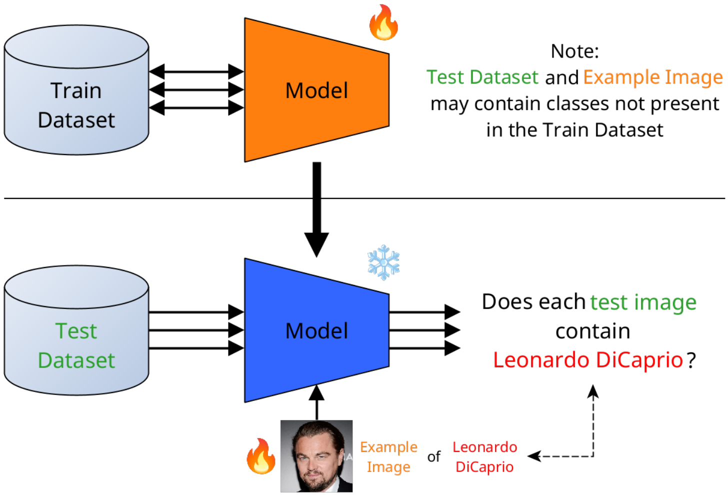
Solution: Obtain extra examples **after** training

Some classification tasks may either have **too many** classes or in the worst case, the set of classes may **not be fixed ahead of time**.

One possible workaround for this issue is to accept *one* (or *a few*) example images for each new class **at inference** / **evaluation time**.

This approach is sometimes called *one*-shot (or *few*-shot) learning.

One-shot / few-shot learning



Outline

1. Introduction
    1.1. Domains and datasets
    1.2. Evaluation and metrics

2. Metric learning methods
    2.1. Sample-based methods
    2.2. Proxy-based methods

3. Efficient searching

Traffic signs

Russian Traffic Sign Images Dataset (RTSD)

105k images, 205 classes of which:
106 classes present in both train and test,
99 classes only available in the test set

Additionally, the authors provide multiple synthetic dataset variants utilizing 3D CGI and generative networks for traffic sign inpainting and stylization.

The proposed synthetic dataset generation methods are a more advanced version of what you will be doing in the next homework. Also, check the paper author list below.

Konushin, Faizov, Shakhuro. Road images augmentation with synthetic traffic signs using neural networks. Computer Optics 2021

Retail products

2000 Retail Product Dataset (RP2k)

10k shelf images,
350k individual product images,
2k different products types

Shelf images collected from 500 stores across 10 cities, extra annotations including product name, brand, type, shape, size and flavour are available

Peng et al. RP2K: A Large-Scale Retail Product Dataset for Fine Grained Image Classification. arXiv 2006

Humans silhouette re-identification

Multi-Scene Multi-Time Person ReID Dataset (MSMT17)

126k bounding boxes,
4k different individuals,
very high data diversity

Collected from 15 different cameras,
over 4 different days in a month,
during 3 different hour intervals

Wei et al. Person Transfer GAN to Bridge Domain Gap for Person Re-Identification. CVPR 2018

Human face recognition

Labeled Faces in the Wild Dataset (LFW)

13k face images, 5k different individuals

Images automatically collected from news photographs. Detect, crop and rescale each face (multiple per image). Manually annotate each face, referencing original news captions.

Originally contained train and test splits, but currently often used purely as a testing dataset

Huang et al. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report 2008

Human face recognition

WebFace42M Dataset (WF42M)

42M face images, 2M different individuals

First, semi-automatically collect celebrity *names* from Freebase, IMDB, etc. Then, scrape images from the internet by using search engines (Google, Bing). Finally, thoroughly clean the data.

There is also a 260M version of the dataset, but it contains raw low quality images with a lot of annotation errors, so almost no one uses it.

Zhu et al. WebFace260M: A Benchmark for Million-Scale Deep Face Recognition. TPAMI 2022

Near-duplicates

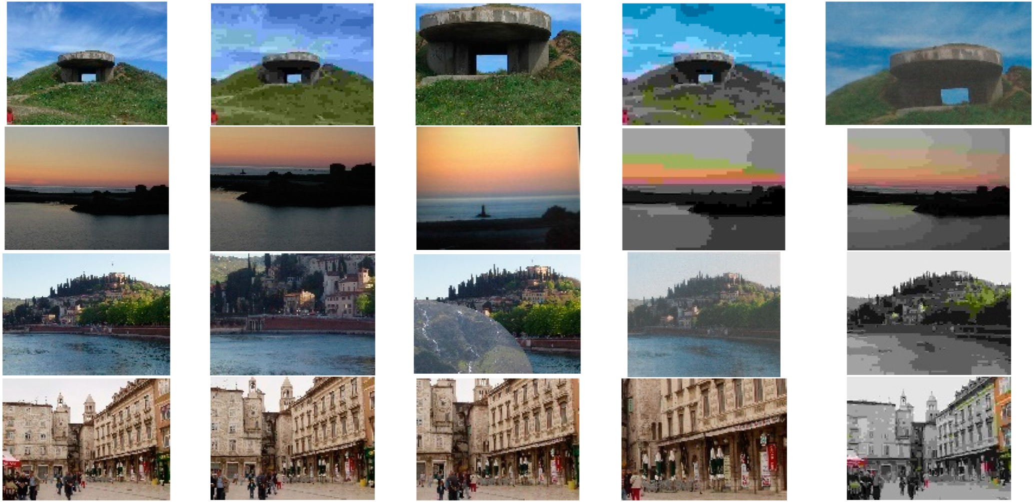


Image search / retrieval

Oxford Landmarks Dataset (Oxford5k)

5k images with Oxford landmarks, 1024×768 resolution

100K and 1M distractor images

Test queries: 5 images per each of 11 landmarks

Philbin et al. Object retrieval with large vocabularies and fast spatial matching. CVPR 2007

Image search / retrieval

Google Landmarks Dataset (GLDv2)

762k index images, 4.1M train images, 200k landmarks

Sourced from Wikimedia, semi-automatic relabelling, 800 human hours

Test queries: 118k images

Weyand et al. Google Landmarks Dataset v2. CVPR 2020

Outline

1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

3. Efficient searching

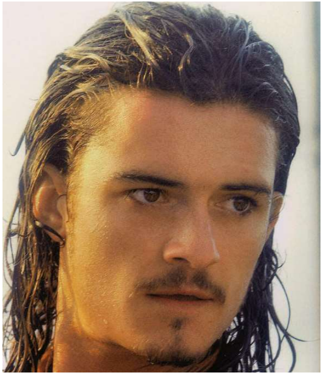
Face Recognition

For simplicity, we will assume Face Recognition (FR) as the default domain for the rest of the lecture, unless stated otherwise.

Most of the metrics, methods and other details discussed here apply equally well to other domains. The names and exact formulations of some metrics might differ from domain to domain.

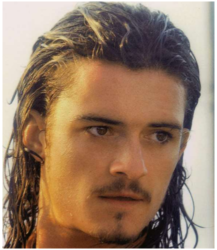
The names of some metric learning methods include explicit references to "Faces", but none of these methods are actually FR-specific. They are widely used in all discussed domains.

Verification (1:1)



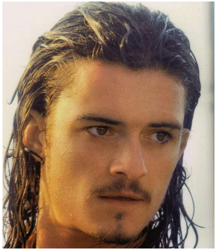
Equivalent pseudo-classification task:
Are these two images of the same person?

Verification (1:1)



Which FR verification applications are you familiar with?

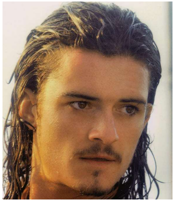
Verification (1:1)

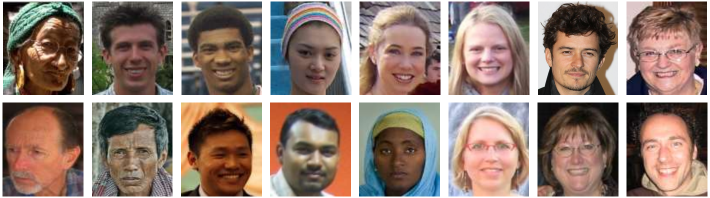


Which FR verification applications are you familiar with?

- unlocking your phone or laptop
- two-factor authentication in banks or government offices
- visa / passport self-verification kiosks on the border
- pay-by-face **(but only as a second factor)**

Identification (1:N)

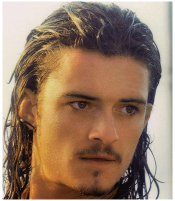


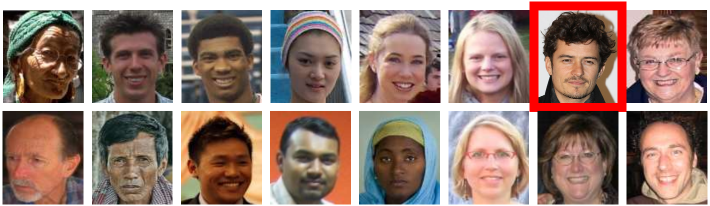


Equivalent pseudo-classification task:

Given a single query image and an enrollment database of N images, determine if the person in the query image is present in the database.

Identification (1:N)

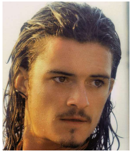


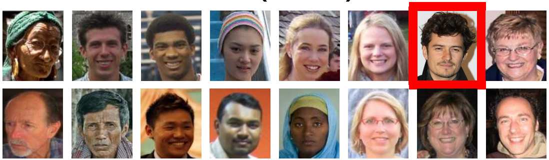


Equivalent pseudo-classification task:

Given a single query image and an enrollment database of N images, determine if the person in the query image is present in the database.

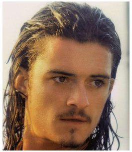
Identification (1:N)

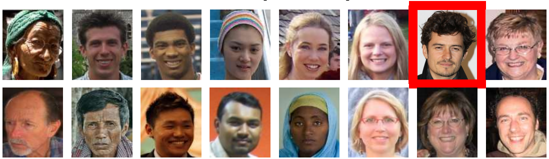




Which FR identification applications are you familiar with?

Identification (1:N)

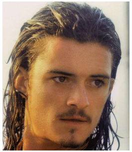


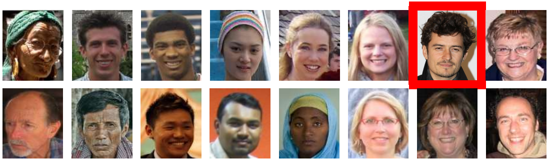


Which FR identification applications are you familiar with?

- white or allow list
  - intercom for residents or entry to restricted / staff-only area
  - pay-by-face (hands free, without your credit card or phone)
- black or deny list
  - law enforcement investigations ("that's Jason Bourne")
- open or dynamic list, **re-**identification
  - customer journey analysis
  - crowd congestion control
  - traffic metrics

Identification (1:N)





Cooperative vs Passive vs Uncooperative

- is the head oriented straight towards the camera?
- is the subject looking at the camera, are their eyes open?
- is the face fully inside the image frame or is it cropped?
- is the face occluded by something (sunglasses, scarf, mask)?

Identification (1:N) is *usually* **cooperative** for white list, **uncooperative** for black list and **passive** for open list. Verification (1:1) is *almost always* **cooperative**.

Classification metrics

		Predicted condition	
	Total population P + N	Predicted Positive	Predicted Negative
Actual condition	Positive P	True Positive TP	False Negative FN
	Negative N	False Positive FP	True Negative TN

Classification metrics

		Predicted condition			
	Total population $P + N$	**Predicted Positive**	**Predicted Negative**		
Actual condition	**Positive** P	**True Positive** TP	**False Negative** FN		**False Negative Rate** $FNR = \frac{FN}{P}$
	Negative N	**False Positive** FP	**True Negative** TN	**False Positive Rate** $FPR = \frac{FP}{N}$	
		Precision $\frac{TP}{TP + FP}$			

Classification metrics

		Predicted condition			
	Total population $P + N$	**Predicted Positive**	**Predicted Negative**		
Actual condition	**Positive** P	**True Positive** TP	**False Negative** FN	**True Positive Rate, Recall, Sensitivity** $TPR = \frac{TP}{P} = 1 - FNR$	**False Negative Rate** $FNR = \frac{FN}{P} = 1 - TPR$
	Negative N	**False Positive** FP	**True Negative** TN	**False Positive Rate** $FPR = \frac{FP}{N} = 1 - TNR$	**True Negative Rate, Specificity, Selectivity** $TNR = \frac{TN}{N} = 1 - FPR$
		Precision $\frac{TP}{TP + FP}$			

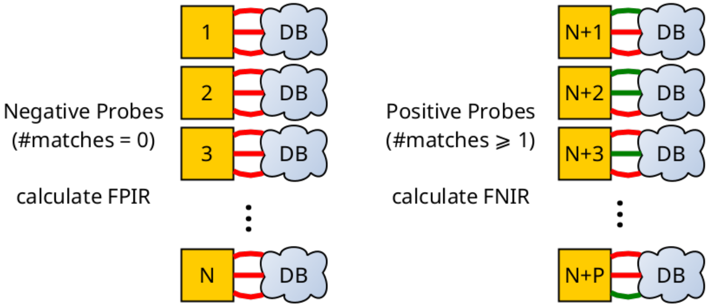
Classification metrics

		Predicted condition			
	Total population $P + N$	**Predicted Positive**	**Predicted Negative**		
Actual condition	**Positive** P	**True Positive** TP	**False Negative** FN	**True Positive Rate, Recall, Sensitivity** $TPR = \frac{TP}{P} = 1 - FNR$	**False Negative Rate** $FNR = \frac{FN}{P} = 1 - TPR$
	Negative N	**False Positive** FP	**True Negative** TN	**False Positive Rate** $FPR = \frac{FP}{N} = 1 - TNR$	**True Negative Rate, Specificity, Selectivity** $TNR = \frac{TN}{N} = 1 - FPR$
		Precision $\frac{TP}{TP + FP}$			

Positive ≈ Match ≈ Acceptance ⇒ FPR = FMR = FAR ≠ FPIR

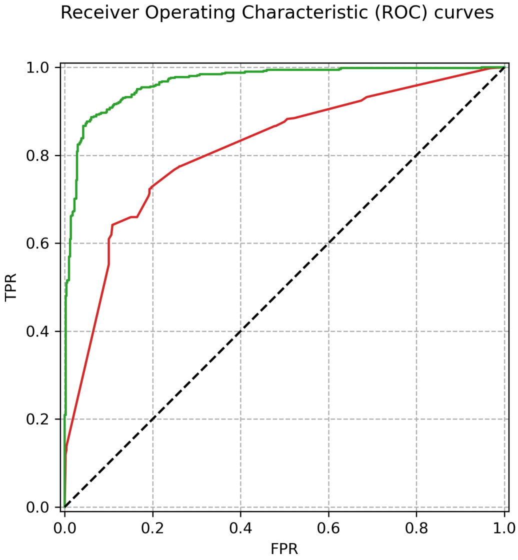
Negative ≈ Non-Match ≈ Rejection ⇒ FNR = FNMR = FRR ≠ FNIR

False Positive / Negative **Identification** Rate

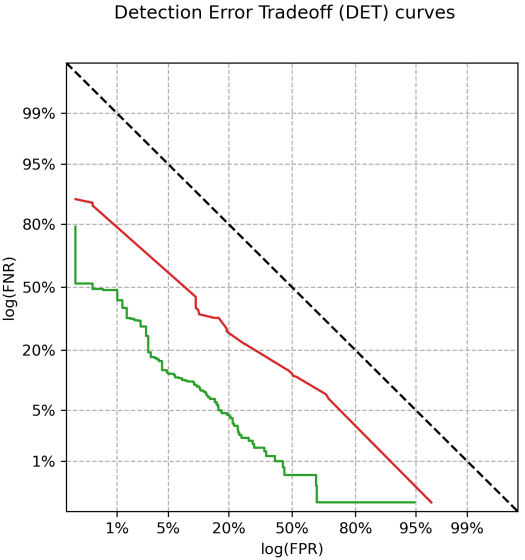


FPIR and FNIR depend on database size (larger DB → harder task)

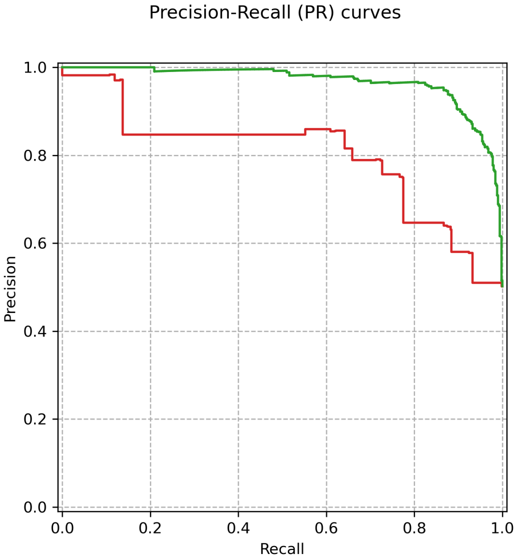
Trade-off curves (V_1 vs V_2)



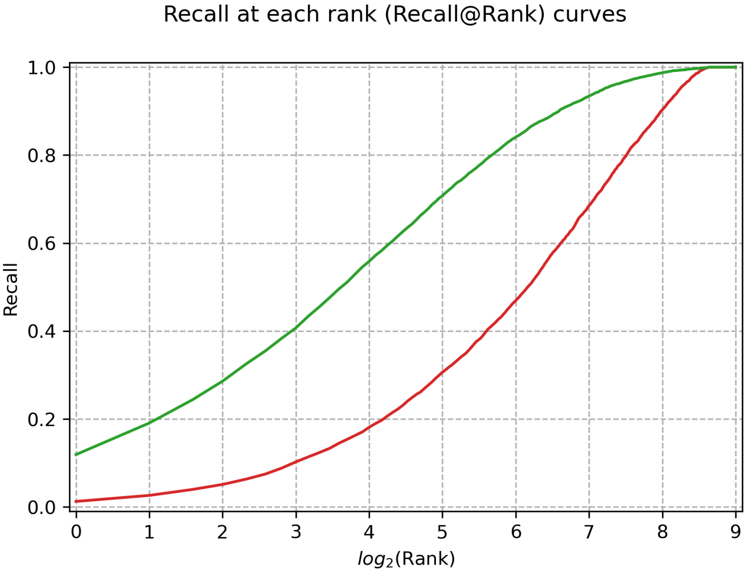
Trade-off curves (V_1 vs V_2)



Trade-off curves (V_1 vs V_2)



Trade-off curves (V_1 vs V_2)



Single point metrics ($V_1@V_2$=const)

Evaluate any of the mentioned trade-off curves at a single point.
For example:

- $\mathrm{TPR@FPR}=10^{-4}$
- $\mathrm{FNIR@Rank}=10$
- $\mathrm{Recall@Rank}=10$
- etc

This approach makes sense, if the value we are fixing represents a realistic use case for the algorithm. So in the above examples, we are checking the performance of our algorithms under the assumption that we can tolerate 1 in 10,000 false positive results ($\mathrm{FPR}=10^{-4}$) or that the user is willing to investigate the top 10 candidates suggested by our algorithms ($\mathrm{Rank}=10$).

Integrated metrics ($\int V_1 \, \mathrm{d}V_2$)

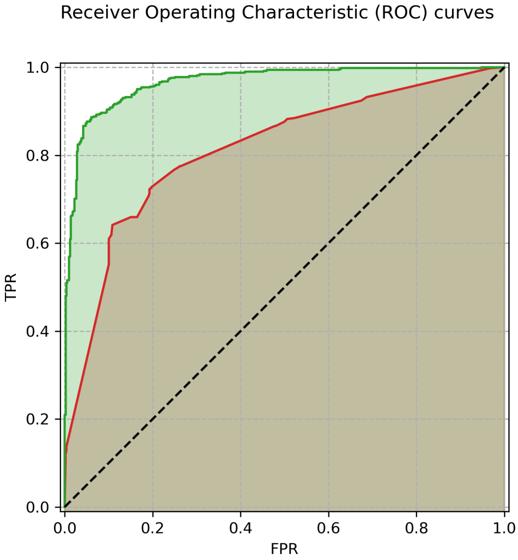
Area Under Curve

- ROC AUC
  (sometimes called just "AUC")
- Precision-Recall AUC
  (sometimes "AUPRC")

Averaged

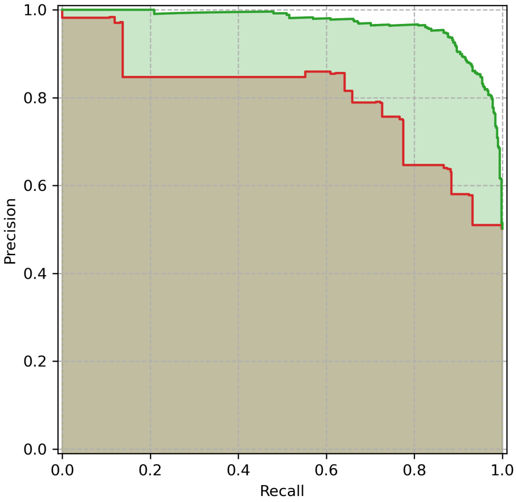
- Average Precision
- Average of any
  "Single point metric"
  at multiple points

Integrated metrics ($\int V_1 \, \mathrm{d}V_2$)



Integrated metrics ($\int V_1 \, \mathrm{d}V_2$)

Precision-Recall (PR) curves



Outline

1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

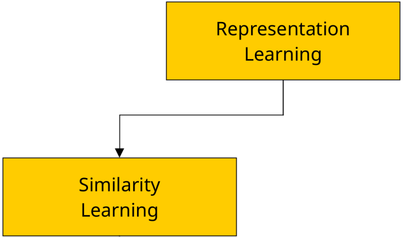
2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

3. Efficient searching

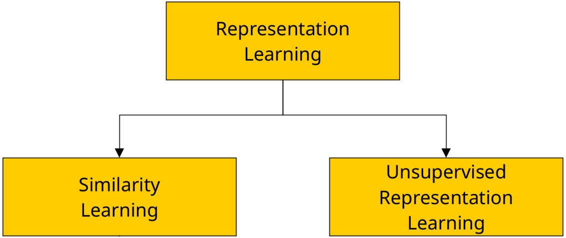
Hierarchy of methods

Representation Learning

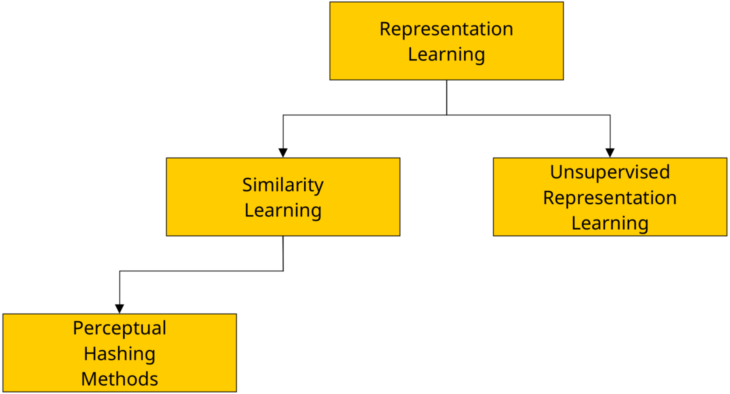
Hierarchy of methods



Hierarchy of methods



Hierarchy of methods



Perceptual hashing methods

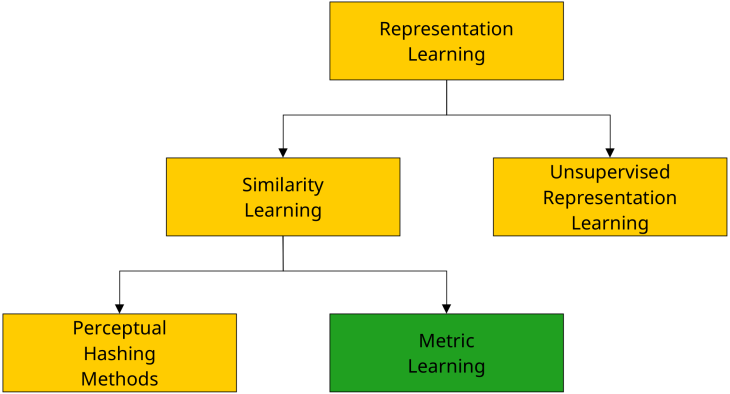
Hand crafted “classical CV” methods. Simple transformations are applied to the images in order to discard perceptually unimportant information (similar to compression algorithms). In the end, the image is reduced to a sequence of bits called its “perceptual hash”.

Generally outside the scope of the course. For the curious students, this blog covers a couple of the simpler methods:

`https://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html`

`https://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html`

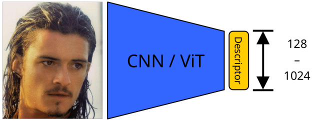
Hierarchy of methods



General metric learning inference pipeline

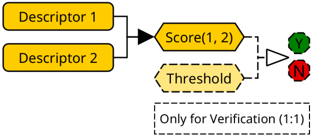
1. Extract Descriptors

aka Embeddings



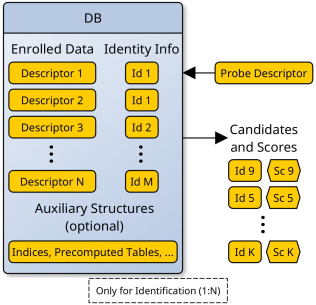
2. Calculate Scores

aka Similarities, aka Distances



3. Search in Database

aka Gallery



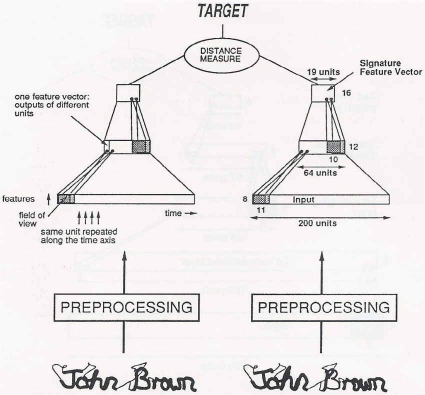
Outline

1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

3. Efficient searching

Siamese networks



Bromley, LeCun et al. Signature verification using a Siamese time delay neural network. NeurIPS 1993

Naive idea: Pull positive pairs, Push negative pairs

$$L = \begin{cases} \left\|f_i - f_j\right\|_2^2 & \text{if } y_i = y_j, \\ -\left\|f_i - f_j\right\|_2^2 & \text{otherwise.} \end{cases}$$

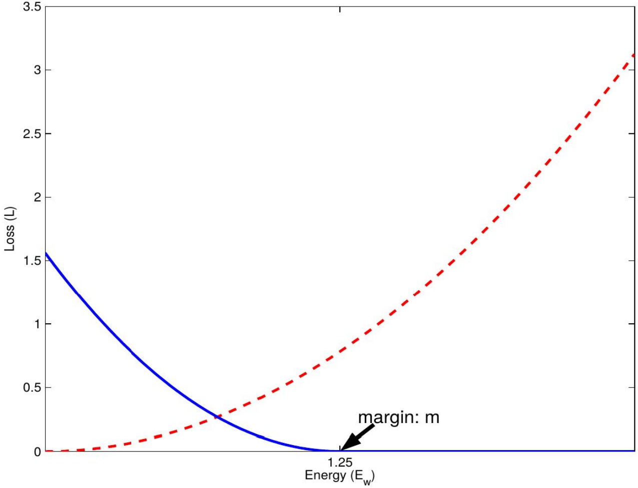
Pairwise contrastive loss

Idea: Sample an **equal number** of positive and negative **pairs** of samples and compute the following loss:

$$L = \begin{cases} \left\| f_i - f_j \right\|_2^2 & \text{if } y_i = y_j, \\ \max\left(0, m - \left\| f_i - f_j \right\|_2\right)^2 & \text{otherwise.} \end{cases}$$

Hadsell, Chopra, LeCun. Dimensionality Reduction by Learning an Invariant Mapping. CVPR 2005

Pairwise contrastive loss



Hadsell, Chopra, LeCun. Dimensionality Reduction by Learning an Invariant Mapping. CVPR 2005

Separability criterion

$$\forall\ f_a, f_p, f_n :$$

$$\left\|f_a - f_p\right\|_2^2 < \left\|f_a - f_n\right\|_2^2$$

Schroff et al. FaceNet: A Unified Embedding for Face Recognition and Clustering. CVPR 2015

Separability criterion

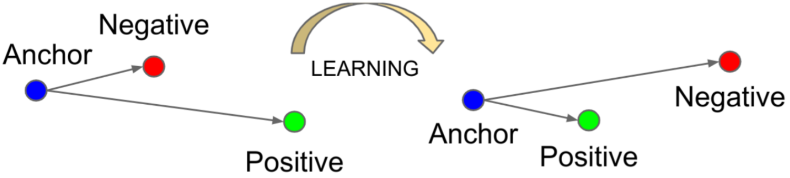
$$\forall \; f_a, f_p, f_n :$$

$$\left\| f_a - f_p \right\|_2^2 + m \; < \; \| f_a - f_n \|_2^2$$

Schroff et al. FaceNet: A Unified Embedding for Face Recognition and Clustering. CVPR 2015

Triplet loss

During training, for each anchor, choose the hardest positive and negative sample.
In particular, choose p with the maximum distance and n with the minimum distance.



$$L = \max\left(0,\ \left\|f_a - f_p\right\|_2^2 + m - \left\|f_a - f_n\right\|_2^2\right)$$

Schroff et al. FaceNet: A Unified Embedding for Face Recognition and Clustering. CVPR 2015

Big problem with sample-based methods

Face Recognition datasets can have 1M+ identities and 100M+ images.

What are the chances that a randomly sampled (or even somewhat smartly selected) batch will contain negative samples that are close/relevant to the chosen anchors?

Sample-based methods rely heavily on good sampling in order to model the whole target embedding space. The complexity of the structure of the embedding space grows with the number of identities/labels that we want to distinguish.

Big problem with sample-based met

Face Recognition datasets can have 1M+ identities and 100M+

What are the chances that a randomly sampled (or even somewhat s cted)
batch will contain negative samples that are close/relevant to the ch a rs?

Sample-based methods rely heavily on good sampling
in order to model the whole target embedding space.
The complexity of the structure of the embedding space grows
with the number of identities/labels that we want to distinguish.

Outline

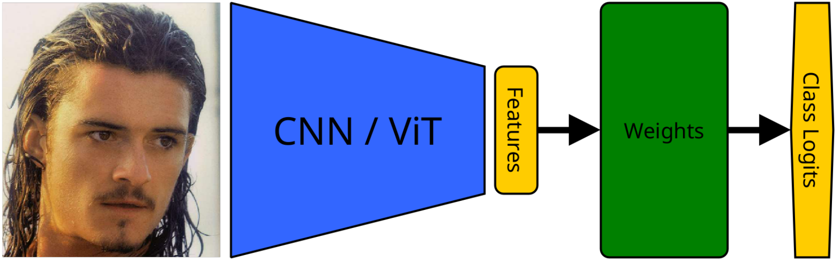
1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

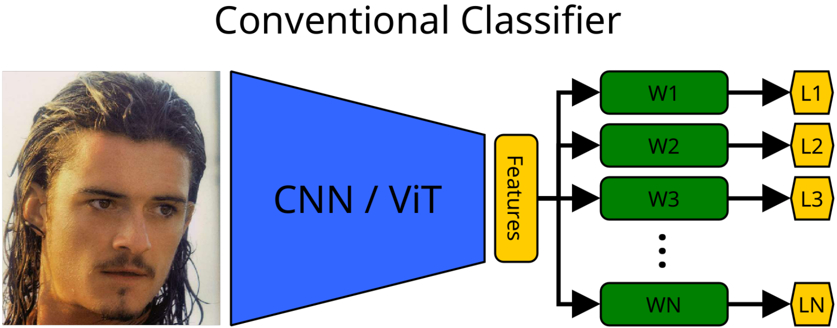
3. Efficient searching

Why shouldn't we just train a classifier?

Conventional Classifier



Why shouldn't we just train a classifier?



Where w_i is the prototype or centroid of the ith class or identity

Is dot product a metric?

$$x^T y = \langle x,\ y \rangle = \|x\|_2\, \|y\|_2 \cos\left(\angle xy\right)$$

Close, but not quite. Negative dot product $-\langle x,\ y \rangle$ behaves *kind of* like a metric (smaller values $\Leftrightarrow$ closer vectors), but it doesn't actually satisfy all the required axioms.

In particular, for any non-zero x and y, you can always arbitrarily increase / decrease $\langle x,\ y \rangle$ by increasing / decreasing the magnitude of $\|x\|_2$ or $\|y\|_2$.

Cosine similarity

This issue can be fixed by applying L_2 normalization to both vectors.
The resulting quantity is called the cosine similarity.

$$\mathrm{c}\left(x,\ y\right) = \frac{x^T y}{\|x\|_2 \, \|y\|_2} = \cos\left(\angle x y\right)$$

Yann LeCun knew this in 1993

The desired output is for a small angle between the outputs of the two subnetworks (f_1 and f_2) when two genuine signatures are presented, and a large angle if one of the signatures is a forgery. For the cosine distance used here:

$$(f_1 \cdot f_2)/(|f_1||f_2|),$$

the desired outputs were 1.0 for a genuine pair of signatures and -0.9 or -1.0 for the second case.

Bromley, LeCun et al. Signature verification using a Siamese time delay neural network. NeurIPS 1993

Cosine distance

Cosine "distance" $\mathrm{d}(x,\ y) = 2 - 2\,\mathrm{c}(x,\ y)$ is technically still not a valid distance metric, but you can prove that optimizing $\mathrm{c}(x,\ y)$ is equivalent to optimizing $\|x - y\|_2^2$, with x and y constrained to the unit hypersphere.

The proof of this fact is left as an exercise to the listener.

SphereFace, CosFace

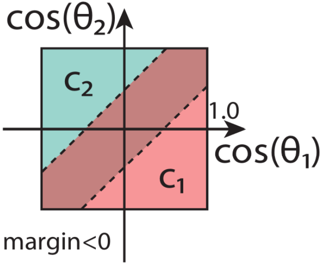
$$L = -\log\left(\frac{e^{s\cdot\left(\mathrm{c}\left(f_i, w_{y_i}\right)-m\right)}}{e^{s\cdot\left(\mathrm{c}\left(f_i, w_{y_i}\right)-m\right)} + \sum_{j\neq y_i} e^{s\cdot\mathrm{c}\left(f_i, w_j\right)}}\right)$$

$$m \sim 0.35$$

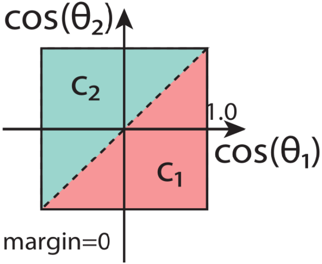
Liu et al. SphereFace: Deep Hypersphere Embedding for Face Recognition. CVPR 2017
Wang et al. CosFace: Large Margin Cosine Loss for Deep Face Recognition. CVPR 2018

SphereFace, CosFace

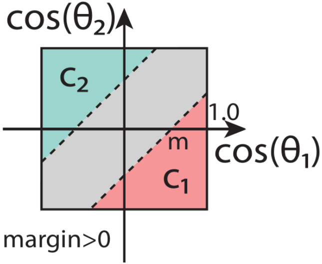
Softmax
Classifier



Softmax
Normalized



CosFace
with Margin



Liu et al. SphereFace: Deep Hypersphere Embedding for Face Recognition. CVPR 2017
Wang et al. CosFace: Large Margin Cosine Loss for Deep Face Recognition. CVPR 2018

xFace

$$L = -\log\left(\frac{e^{s\cdot \mathrm{T}\left(\theta_{y_i}\right)}}{e^{s\cdot \mathrm{T}\left(\theta_{y_i}\right)} + \sum_{j\neq y_i} e^{s\cdot \mathrm{c}\left(f_i, w_j\right)}}\right)$$

where $\theta_j = \arccos\left(\mathrm{c}\left(f_i, w_j\right)\right)$

Zhang et al. Unifying Margin-Based Softmax Losses in Face Recognition WACV 2023

xFace, $x \in \{$Sphere, Cos, Arc, Amp$\}$

$$\mathrm{T}(\theta) = m_0 \cdot \cos\left(m_1 \cdot \theta + m_2\right) - m_3$$

SphereFace: $m_1 \sim 1.35$

CosFace: $m_3 \sim 0.35$

ArcFace: $m_2 \sim 0.5$

AmpFace: $m_0 \sim 0.375$

Zhang et al. Unifying Margin-Based Softmax Losses in Face Recognition WACV 2023

Big problem with sample-based met

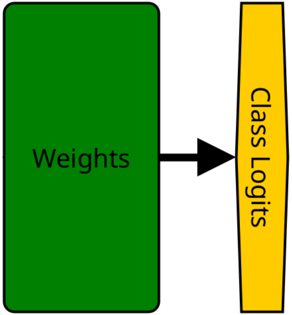
Face Recognition datasets can have 1M+ identities and 100M+

What are the chances that a randomly sampled (or even somewhat s cted) batch will contain negative samples that are close/relevant to the ch a rs?

Sample-based methods rely heavily on good sampling in order to model the whole target embedding space. The complexity of the structure of the embedding space grows with the number of identities/labels that we want to distinguish.

Even bigger problem with proxy-based methods

Let's estimate the size of the weight/centroid/prototype matrix W and the logits tensor O:



$$W \in \mathbb{R}^{C \times E}, \quad O \in \mathbb{R}^{B \times C}$$

even for somewhat small values of $C \sim 1\text{M}$, $E \sim 1024$, $B \sim 1024$, we get

$$\text{sizeof}(W) = C \cdot E \cdot \text{sizeof}(\text{fp32}) \sim 4\text{GB}$$

$$\text{sizeof}(O) = B \cdot C \cdot \text{sizeof}(\text{fp32}) \sim 4\text{GB}$$

Additionally, the following tensors may require similarly sized allocations:

- intermediate computations involving O during the forward pass
- intermediate computations involving $\frac{dL}{dO}$ during the backward pass
- gradient storage/accumulator for $\frac{dL}{dW}$ during the backward pass
- storage for additional optimizer state for W (e.g. `exp_avg` and `exp_avg_sq` for Adam)

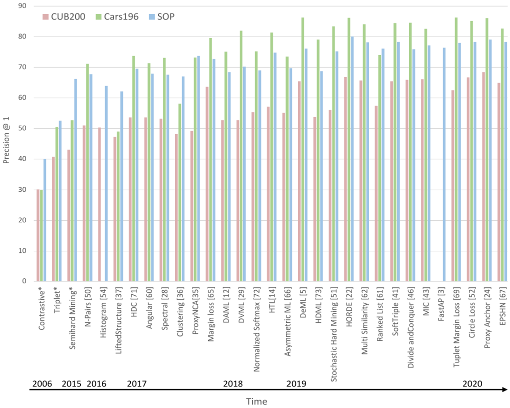
And that is not counting the memory requirements for training the extractor model itself!

Comparison on Megaface benchmark

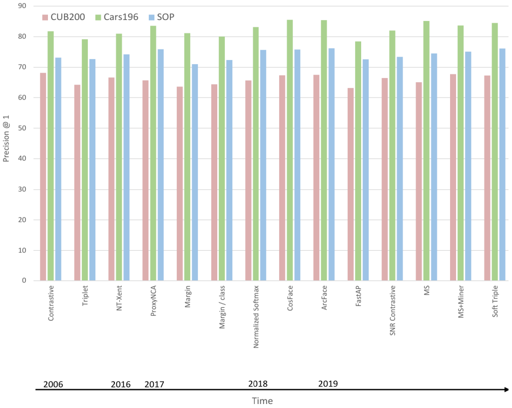
Methods	Id (%)	Ver (%)
Softmax [15]	54.85	65.92
Contrastive Loss[15, 30]	65.21	78.86
Triplet [15, 27]	64.79	78.32
Center Loss[36]	65.49	80.14
SphereFace [15]	72.729	85.561
CosFace [35]	77.11	89.88
AM-Softmax [33]	72.47	84.44
SphereFace+ [14]	73.03	-
CASIA, R50, ArcFace	77.50	92.34
CASIA, R50, ArcFace, R	91.75	93.69
FaceNet [27]	70.49	86.47
CosFace [35]	82.72	96.65
MS1MV2, R100, ArcFace	81.03	96.98
MS1MV2, R100, CosFace	80.56	96.56
MS1MV2, R100, ArcFace, R	98.35	98.48
MS1MV2, R100, CosFace, R	97.91	97.91

Table 6. Face identification and verification evaluation of different methods on MegaFace Challenge1 using FaceScrub as the probe set. "Id" refers to the rank-1 face identification accuracy with 1M distractors, and "Ver" refers to the face verification TAR at 10^{-6} FAR. "R" refers to data refinement on both probe set and 1M distractors. ArcFace obtains state-of-the-art performance under both small and large protocols.

A metric learning reality check



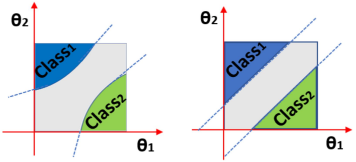
(a) The trend according to papers



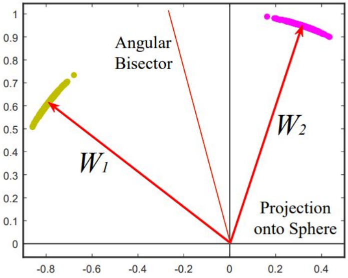
(b) The trend according to reality

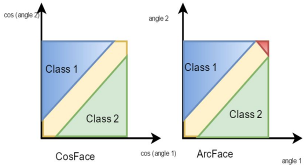
Musgrave et al. A metric learning reality check. ECCV 2020

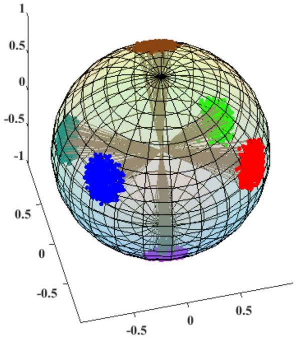
Beware of pretty plots

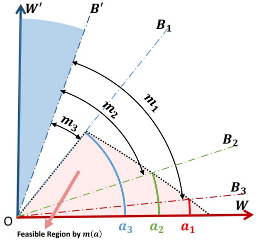


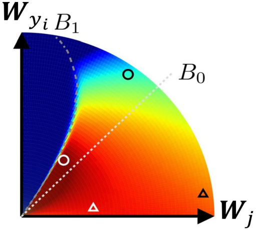
CosFace ArcFace



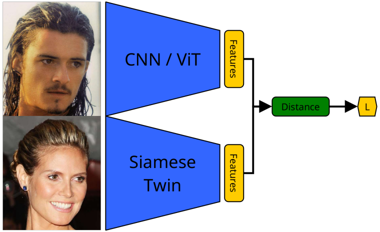




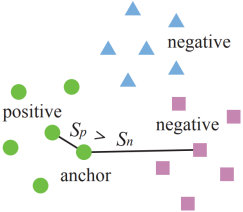


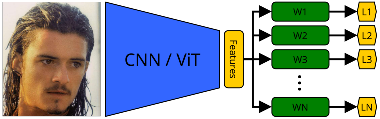


Unfair comparisons

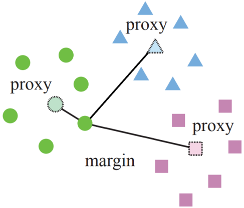


$$L = \max\left(0,\ \left\|f_a - f_p\right\|_2^2 + m - \left\|f_a - f_n\right\|_2^2\right)$$





$$L = -\log\left(\frac{e^{s\cdot\left(c\left(f_i, w_{y_i}\right)-m\right)}}{e^{s\cdot\left(c\left(f_i, w_{y_i}\right)-m\right)} + \sum_{j\neq y_i} e^{s\cdot c\left(f_i, w_j\right)}}\right)$$



Smooth-AP OneFace
In Defense of the Triplet Loss
TransFace
UniFace
AnchorFace Entropy-guided Hard Sample Mining
Pairwise Similarity Learning is SimPLE
Consistent Instance False Positive
Sampling Matters
SphereFace2
Hidden Pitfalls of the Cosine Similarity
Multi-Similarity
Discrepancy Alignment Metric
Tuplet Margin UniTSFace
Lifted Structured Feature Embedding

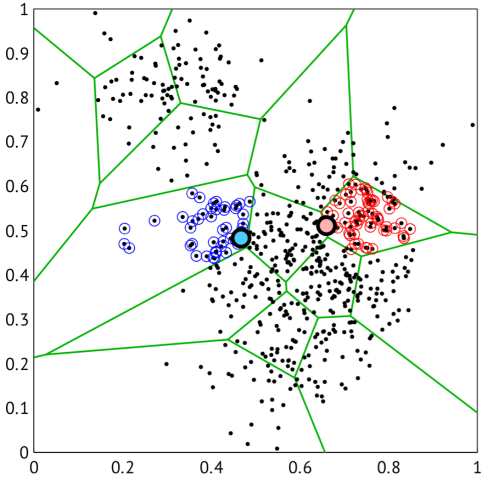
Outline

1. Introduction
   1.1. Domains and datasets
   1.2. Evaluation and metrics

2. Metric learning methods
   2.1. Sample-based methods
   2.2. Proxy-based methods

3. Efficient searching

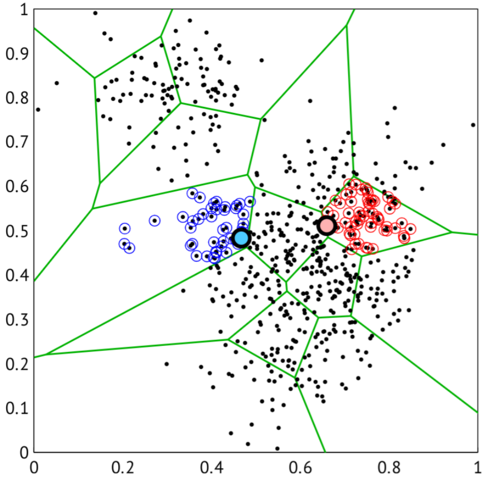
Inverted index



Construction: use k-means to divide images into K clusters. K centroids (*codewords*) form a *codebook*. Store K lists with image ids in RAM

Search: given a query, find several nearest *codewords*. List all elements in resp. clusters

Inverted index

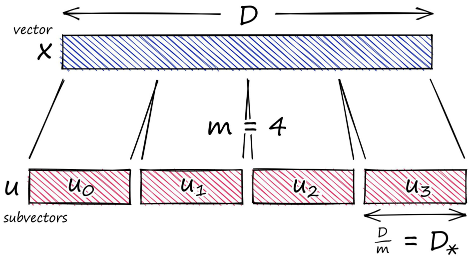


Construction: use k-means to divide images into K clusters. K centroids (*codewords*) form a *codebook*. Store K lists with image ids in RAM

Search: given a query, find several nearest *codewords*. List all elements in resp. clusters

Drawbacks?

Product quantization



Construction: divide vector into m parts, encode each subvector with k-means

Usually $k_* = 256$ (1 byte code per subvector)

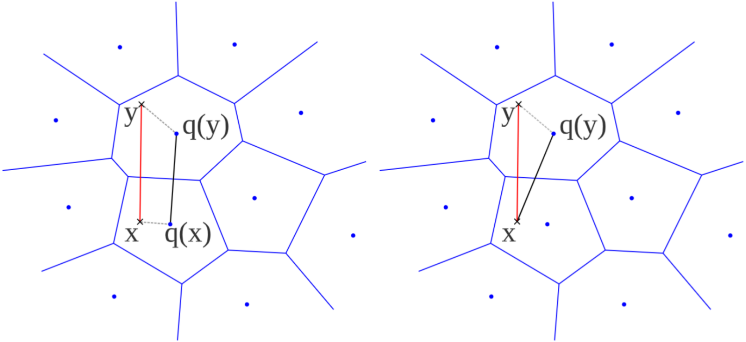
Comparison with k-means
Memory and search complexity:

$$kD \quad \text{vs} \quad mk_*D_* = k^{1/m}D$$

because $D = mD_*$ and $k \approx k_*^m$

(assuming subvectors are independent)

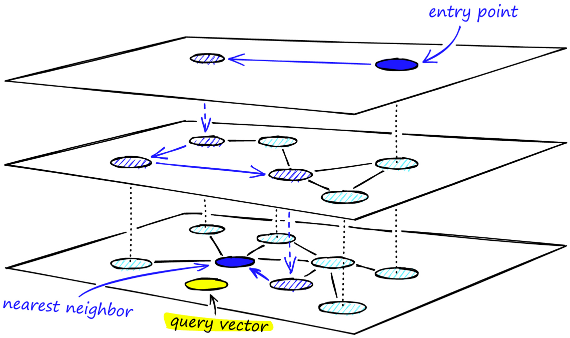
Distance computation in PQ method



symmetric case   asymmetric case

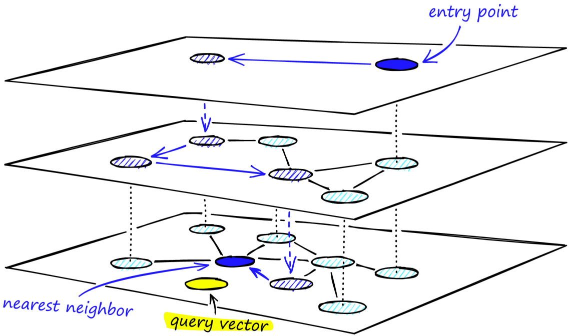
$$\|x-y\|^2 \approx \|x-[q_1(y),\dots,q_m(y)]\|^2 = \sum_{i=1}^{m} \|x_i - q_i(y)\|^2$$

Hierarchical Navigable Small World



Malkov, Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. TPAMI 2018

Hierarchical Navigable Small World



Rough complexity estimates

Search in $O(\log N)$

Construct in $O(N \log N)$

Memory: 60-450 bytes/object

Malkov, Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. TPAMI 2018

Overall method

1. Compute inverted index with large K $= 2^{20}$
2. In each cluster encode residual vectors with PQ
3. Use HNSW to choose clusters during search

		DEEP1B					SIFT1B				
Method	K	R@1	R@10	R@100	t	Mem	R@1	R@10	R@100	t	Mem
O-Multi-D-OADC[24]	2^{14}	0.397	0.766	0.909	8.5	17.34	0.360	0.792	0.901	5	17.34
Multi-LOPQ[4]	2^{14}	0.41	0.79	-	20	18.68	**0.454**	**0.862**	0.908	19	19.22
GNOIMI[5]	2^{14}	0.45	0.81	-	20	19.75	-	-	-	-	-
IVFOADC+G+P	2^{20}	**0.452**	**0.832**	**0.947**	**3.3**	17.87	0.405	0.851	**0.957**	**3.5**	18

Table 4. Comparison to the previous works for 16-byte codes. The search runtimes are reported in milliseconds. We also provide the memory per point required by the retrieval systems (the numbers are in bytes and do not include 4 bytes for point ids).

Baranchuk et al. Revisiting the inverted indices for billion-scale approximate nearest neighbors. ECCV 2018

Conclusion

We reviewed following topics:

- pseudo-classification tasks across different domains
- relevant practical applications and metrics
- sample-based and proxy-based metric learning methods
- several approximate nearest neighbour methods for faster search and indexing in metric representation spaces