34º Escuela de Ciencias Informáticas
Departamento de Computación, FCEyN, UBA

Aprendizaje con datos escasos

Jorge Sánchez
CIEM-CONICET, UNC
jorge.sanchez@unc.edu.ar

Plan del curso

1. Introducción. Problemas. Representaciones en visión y lenguaje
2. Aprendizaje por transferencia y aumento de datos
3. Aprendizaje basado en prototipos
4. **Aprendizaje de métricas**
5. Aprendizaje sin ejemplos, auto supervisión, etc

Metric learning

Distance Functions

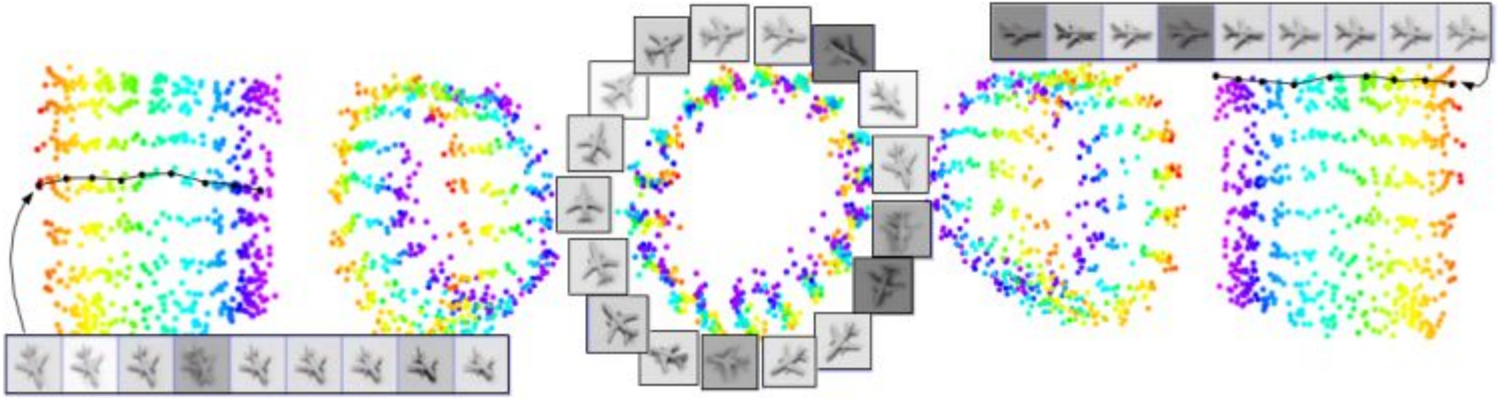
The concept of distance function $d(.,.)$ is **inherent** to any pattern recognition problem. E.g. clustering (kmeans), classification (kNN, SVM) etc.

Typical Choices

- Minkowski Distance: $L_p(P,Q) = (\sum_i |P_i - Q_i|^p)^{\frac{1}{p}}$.
- Cosine: $L(P,Q) = \frac{P^T Q}{|P||Q|}$
- Earth Mover: Uses an optimization algorithm
- Edit distance: Uses dynamic programming between sequences.
- KL Divergence: $KL(P \parallel Q) = \sum_i P_i \log \frac{P_i}{Q_i}$. (Not Symmetric!)
- many more ... (depending on type of problem)

Distance Metric Learning

Learn a function that maps input patterns into a target space such that the simple distance in the target space (Euclidean) approximates the "semantic" distance in the input space.



slides: Praveen Krishnan

What defines a metric?

1. Non-negativity: $D(P, Q) \geq 0$
2. Identity of indiscernibles: $D(P, Q) = 0$ iff $P = Q$
3. Symmetry: $D(P, Q) = D(Q, P)$
4. Triangle Inequality: $D(P, Q) \leq D(P, K) + D(K, Q)$

Pseudo/Semi Metric

If the second property is not followed strictly i.e. "iff $\rightarrow$ if"

Mahalanobis distances

- Assume the data is represented as N vectors of length d: $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$
- Squared Euclidean distance

$$
\begin{aligned}
d(\mathbf{x}_1, \mathbf{x}_2) &= \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \\
&= (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)
\end{aligned}
$$

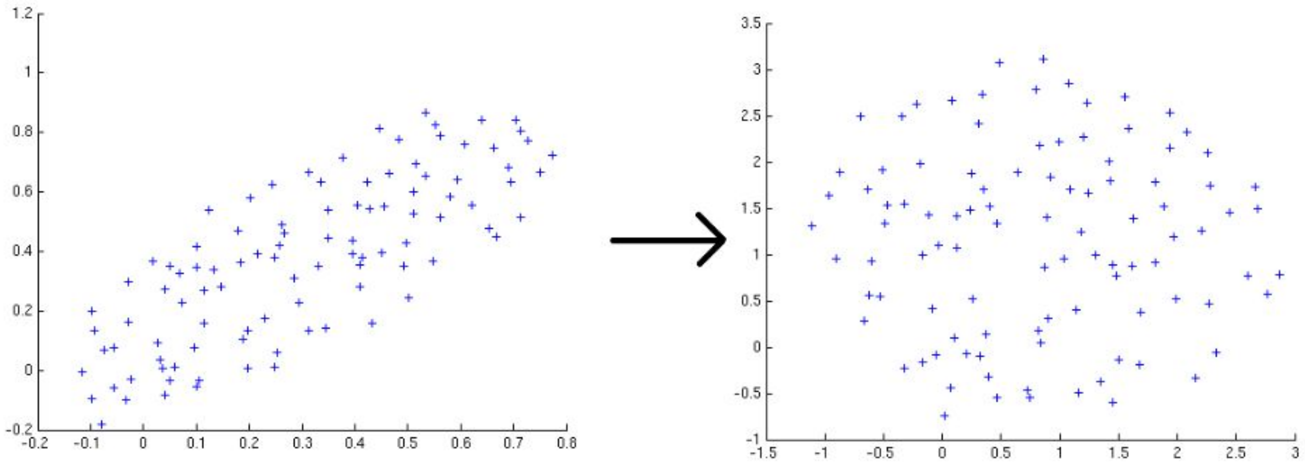
- Let $\Sigma = \sum_{i,j}(\mathbf{x}_i - \mu)(\mathbf{x}_j - \mu)^T$
- The "original" Mahalanobis distance:

$$
d_M(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2)
$$

slides: Brian Kulis

Mahalanobis distances

- Equivalent to applying a *whitening transform*



Mahalanobis distances

- Assume the data is represented as N vectors of length d: $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$
- Squared Euclidean distance

$$\begin{aligned} d(\mathbf{x}_1, \mathbf{x}_2) &= \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \\ &= (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) \end{aligned}$$

- Mahalanobis distances for metric learning
  - Distance parametrized by $d \times d$ positive semi-definite matrix A:

$$d_A(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T A(\mathbf{x}_1 - \mathbf{x}_2)$$

- Used for many existing metric learning algorithms

Mahalanobis distances

$$d_A(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2)$$

- Why is A positive semi-definite (PSD)?
  - If A is not PSD, then d_A could be negative
  - Suppose $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_2$ is an eigenvector corresponding to a negative eigenvalue λ of A

$$\begin{aligned} d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\ &= \mathbf{v}^T A \mathbf{v} \\ &= \lambda \mathbf{v}^T \mathbf{v} = \lambda < 0 \end{aligned}$$

Mahalanobis distances

- Properties of a metric:
  - $d(\mathbf{x}, \mathbf{y}) \geq 0$
  - $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$
  - $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
  - $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$
- d_A is not technically a metric
  - Analogous to Euclidean distance, need the square root:

$$\sqrt{d_A(\mathbf{x}_1, \mathbf{x}_2)} = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T A(\mathbf{x}_1 - \mathbf{x}_2)}$$

- Square root of the Mahalanobis distance satisfies all properties if A is strictly positive definite, but if A is positive semi-definite then second property is not satisfied
  - Called a *pseudo-metric*
- In practice, most algorithms work only with d_A

slides: Brian Kulis

Mahalanobis distances

- Can view d_A as the squared Euclidean distance after applying a linear transformation
  - Decompose $A = G^T G$ via Cholesky decomposition
  - (Alternatively, take eigenvector decomposition $A = V \Lambda V^T$ and look at $A = (\Lambda^{1/2} V^T)^T (\Lambda^{1/2} V^T)$)
- Then we have

$$\begin{aligned} d_A(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2) \\ &= (\mathbf{x}_1 - \mathbf{x}_2) G^T G (\mathbf{x}_1 - \mathbf{x}_2) \\ &= (G\mathbf{x}_1 - G\mathbf{x}_2)^T (G\mathbf{x}_1 - G\mathbf{x}_2) \\ &= \|G\mathbf{x}_1 - G\mathbf{x}_2\|_2^2 \end{aligned}$$

- Mahalanobis distance is just the squared Euclidean distance after applying the linear transformation G

Metric learning problem formulation

- Typically 2 main pieces to a Mahalanobis metric learning problem
  - A set of constraints on the distance
  - A regularizer on the distance / objective function
- In the constrained case, a general problem may look like:

$$\begin{array}{ll} \min_A & r(A) \\ \text{s.t.} & c_i(A) \leq 0 \quad 0 \leq i \leq C \\ & A \succeq 0 \end{array}$$

- r is a regularizer/objective on A and c_i are the constraints on A
- An unconstrained version may look like:

$$\min_{A \succeq 0} r(A) + \lambda \sum_{i=1}^{C} c_i(A)$$

slides: Brian Kulis

Defining constraints

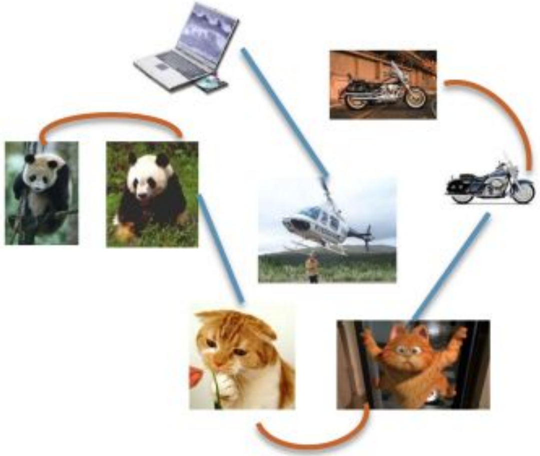
- Similarity / Dissimilarity constraints
  - Given a set of pairs $\mathcal{S}$ of points that should be similar, and a set of pairs of points $\mathcal{D}$ of points that should be dissimilar
  - A single constraint would be of the form

$$d_A(\mathbf{x}_i, \mathbf{x}_j) \leq \ell$$

    for $(i, j) \in \mathcal{S}$ or

$$d_A(\mathbf{x}_i, \mathbf{x}_j) \geq u$$

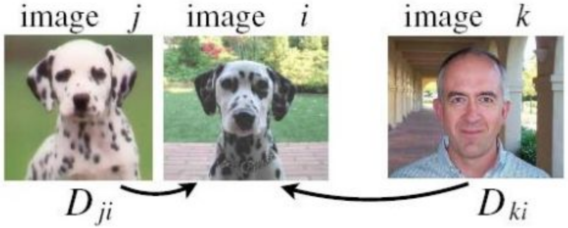
    for $(i, j) \in \mathcal{D}$
  - Easy to specify given class labels



- Relative distance constraints
  - Given a triple $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ such that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_k$, a single constraint is of the form

$$d_A(\mathbf{x}_i, \mathbf{x}_j) \leq d_A(\mathbf{x}_i, \mathbf{x}_k) - m,$$

    where m is the margin
  - Popular for ranking problems

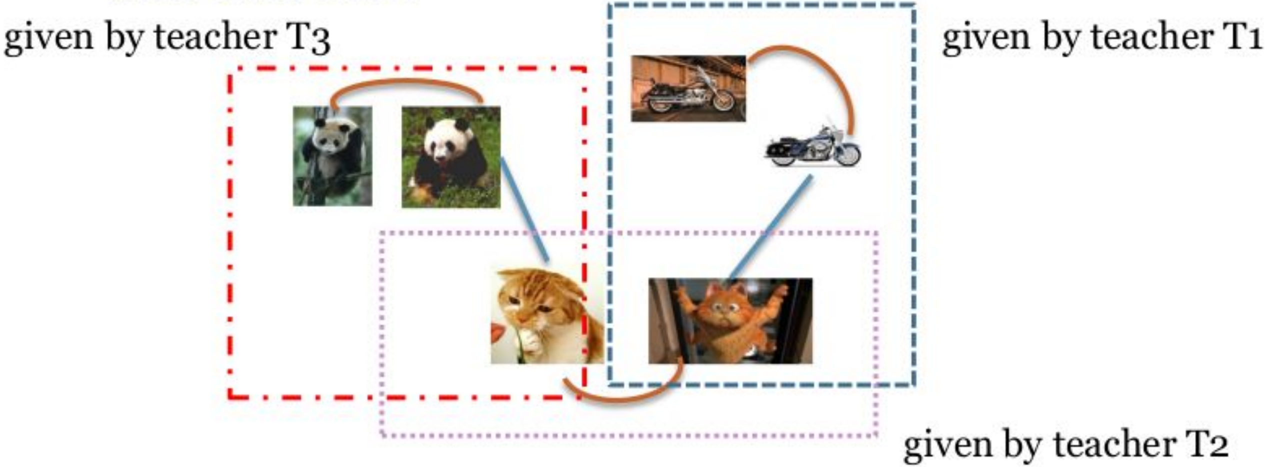


Why not labels?

- Sometimes constraints are easier to get than labels
  - faces extracted from successive frames in a video in roughly the same location can be assumed to come from the same person

Why not labels?

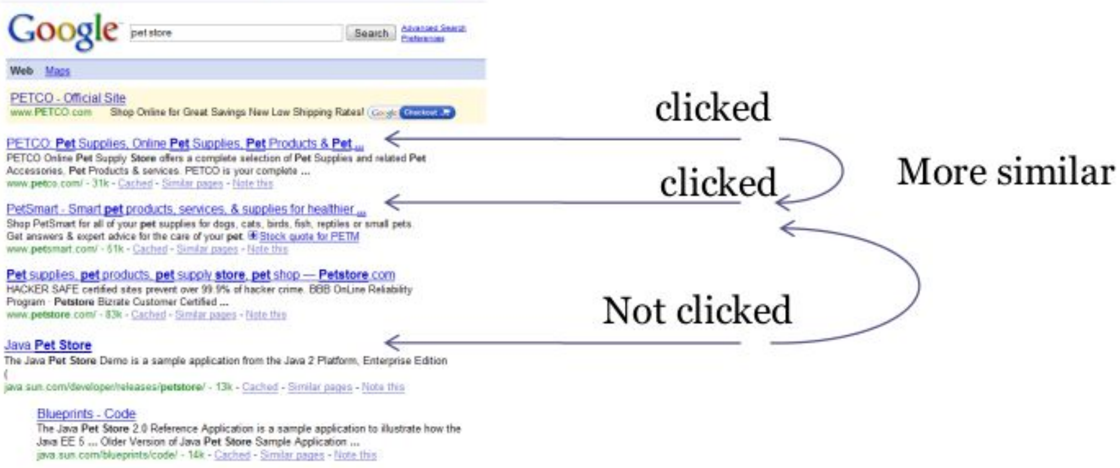
- Sometimes constraints are easier to get than labels
  - Distributed Teaching
    - Constraints are given by teachers who don't coordinate with each other



slides: Xin Sui

Why not labels?

- Sometimes constraints are easier to get than labels
  - Search engine logs



The basic formulation

Attempt: Let's create two sets of pairs: similar set *S*, dissimilar set *D*.

want M such that: $\rho_M(x, x')$ large, for $(x, x') \in D$
$\rho_M(x, x')$ small, for $(x, x') \in S$

Create cost/energy function: $\Psi(M)$

$$\Psi(M) = \lambda \sum_{(x,x') \in S} \rho_M^2(x, x') - (1-\lambda) \sum_{(x,x') \in D} \rho_M^2(x, x')$$

Minimize $\Psi(M)$ with respect to *M*!

slides: Nakul Verma

Mahalanobis Metric for Clustering (MMC)

maximize$_M$ $\displaystyle\sum_{(x,x')\in D} \rho_M^2(x,x')$

constraint: $\displaystyle\sum_{(x,x')\in S} \rho_M^2(x,x') \leq 1$

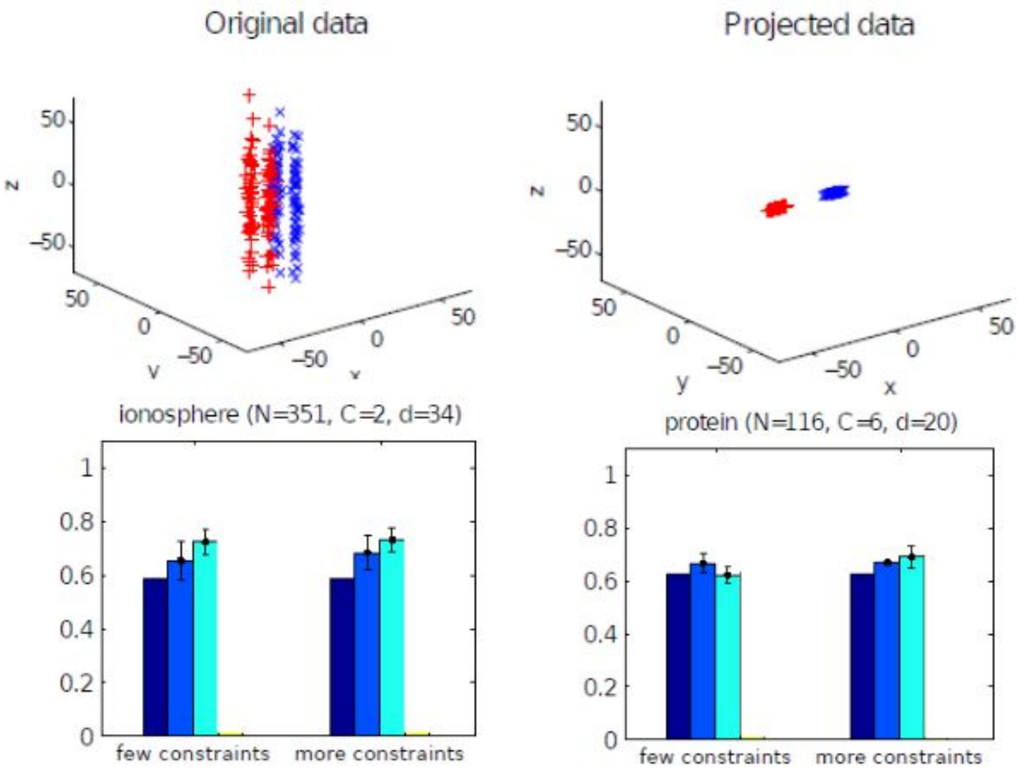
$M \in \mathrm{PSD}$

Recall:
$M = W^\mathsf{T} W$

Advantages:

- Problem formulation is convex, so efficiently solvable!
- Tight convex clusters, can help in clustering!

Mahalanobis Metric for Clustering (MMC)



Left to right: *k*-means, *k*-means+diag MMC, *k*-means + full MMC.

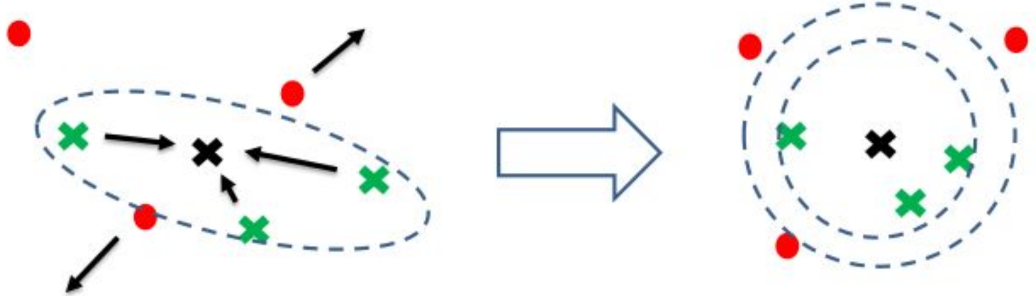
Large Margin Nearest Neighbors (LMNN)

$$\Psi_{\text{pull}}(M) = \sum_{i,j(i)} \rho_M^2(x_i, x_j)$$

$$\Psi_{\text{push}}(M) = \sum_{i,j(i),l(i,j)} \left[1 + \rho_M^2(x_i, x_j) - \rho_M^2(x_i, x_l)\right]_+$$

point	i
true neighbor	$j(i)$
imposter	$l(i,j)$

$[a]_+ = \max\{0, a\}$



$$\Psi(M) = \lambda\ \Psi_{\text{pull}}(M)\ + (1-\lambda)\ \Psi_{\text{push}}(M)$$

Advantages:

- Local constraints, so directly improves nearest neighbor quality!

Wineberger et al. (2006) Distance Metric Learning for Large Margin Nearest Neighbor Classification

slides: Nakul Verma

Large Margin Nearest Neighbors (LMNN)

Dataset	k-NN best	LMNN best	SVM
mnist	2.12	**1.18**	1.20
letters	4.63	**2.67**	3.21
isolet	5.90	**3.40**	**3.40**
yfaces	4.80	**4.05**	15.22
balance	10.82	5.86	**1.92**
wine	2.17	**2.11**	22.24
iris	4.00	3.68	**3.45**

Wineberger et al. (2006) Distance Metric Learning for Large Margin Nearest Neighbor Classification

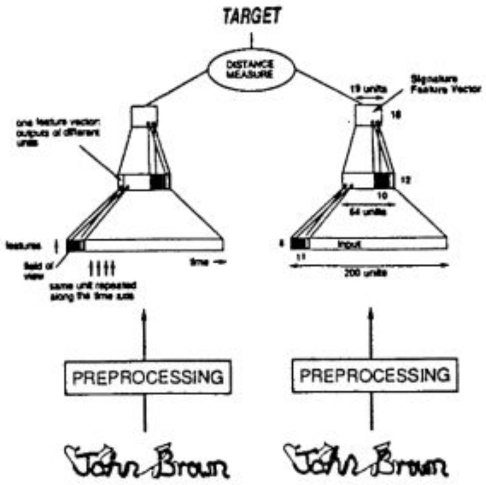
slides: Nakul Verma

Deep metric learning

Siamese networks

Siamese is an informal term for conjoined or fused.

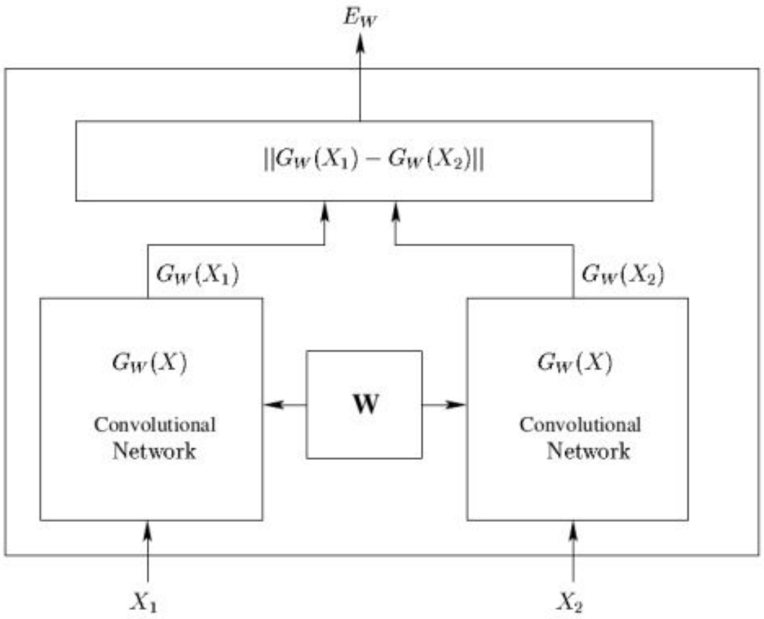
- Contains two or more identical sub-networks with shared set of parameters and weights
- Popularly used for similarity learning tasks such as verification and ranking.



Siamese networks

Given a family of functions $G_W(X)$ parameterized by W, find W such that the similarity metric $D_W(X_1, X_2)$ is small for similar pairs and large for disimilar pairs:-

$$D_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)||$$



slide: Praveen Krishnan

Contrastive loss

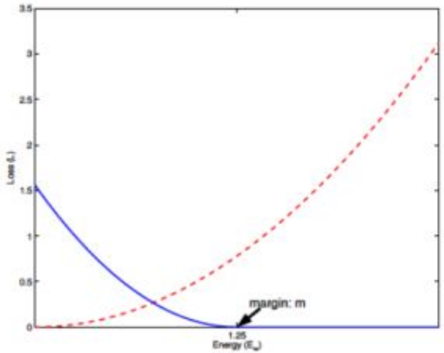
Let $X_1, X_2 \in \mathcal{I}$, pair of input vectors and Y be the binary label where $Y = 0$ means the pair is similar and $Y = 1$ means dissimilar. We define a parameterized distance function D_W as:-

$$D_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)||_2$$

The contrastive loss function is given as:-

$$L(W, Y, X_1, X_2) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

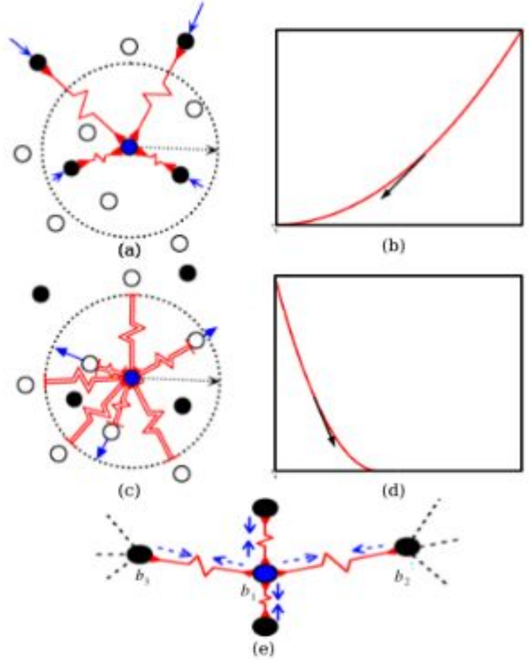
Here $m > 0$ is the margin which enforces the robustness.



slide: Praveen Krishnan

Contrastive loss

Spring model analogy: $F = -KX$



Attraction

$$\frac{\partial L_S}{\partial W} = D_W \frac{\partial D_W}{\partial W}$$

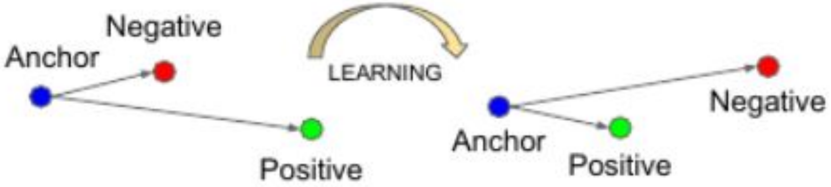
Repulsion

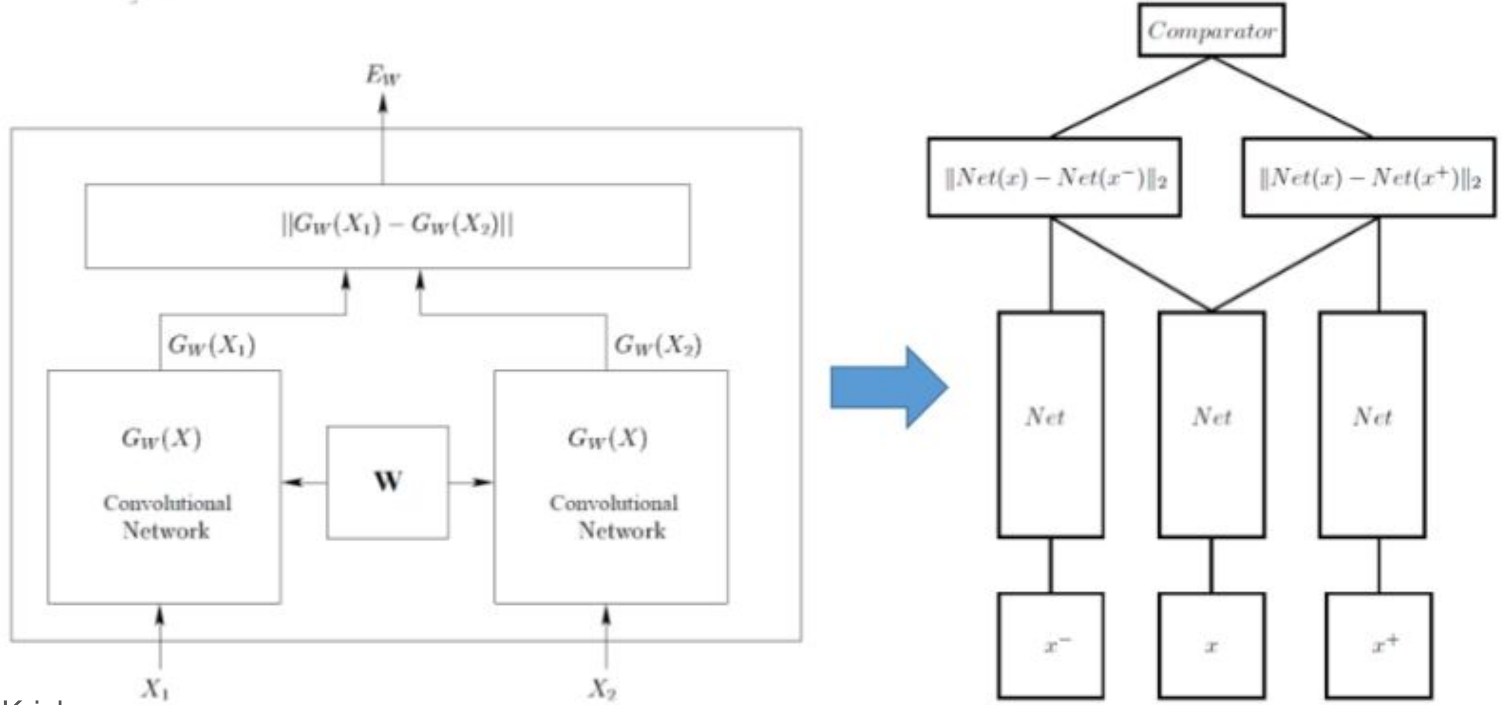
$$\frac{\partial L_D}{\partial W} = -(m - D_W) \frac{\partial D_W}{\partial W}$$

The force is absent when $D_W \geq m$.

slide: Praveen Krishnan

Triplet network



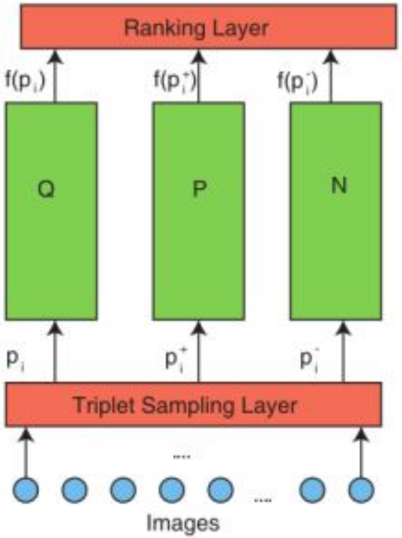


slide: Praveen Krishnan

Triplet loss

- Learn an embedding function $f(.)$ that assigns smaller distances to similar image pairs.
- Given a triplet: $t_i = (p_i, p_i^+, p_i^-)$, triplet loss is defined as:-

$$l(p_i, p_i^+, p_i^-) = max\{0, m + D(f(p_i), f(p+i^+)) - D(f(p_i), f(p_i^-))\}$$



slide: Praveen Krishnan

Triplet mining

Selection of triplets important for faster convergence and better training.

Challenges

- Given N examples, picking all triplets is $\mathcal{O}(N^3)$.
- Need for fresh selection of triplets after each epoch.

Typical Strategies

- Select hard positives and hard negatives.
- Generate triplets offline every n steps 'or' **online** from each mini batch.

slide: Praveen Krishnan

Triplet mining

Schroff et. al. arxiv'15

- Generate triplets online with large mini batch sizes by ensuring minimum no. of exemplars for each class.
- Picks semi-hard examples where:-

$$||G(X_i^a) - G(X_i^p)||_2^2 < ||G(X_i^a) - G(X_i^n)||_2^2$$

These negative samples are further away from anchor but lie inside the margin m

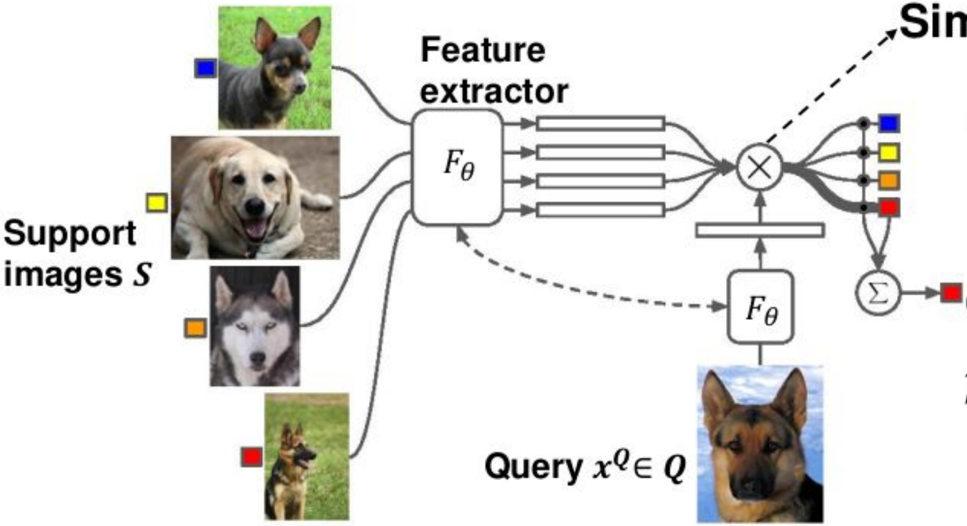
Wang et. al. CVPR'14

- Uses pairwise relevance scores (prior knowledge).
- Uses an online triplet sampling algorithm based on reservoir sampling.

slide: Praveen Krishnan

Matching Networks

- **Learn to match**
  - Extract features from the query and support images
  - Classify with **differentiable (soft) nearest neighbor classifier**



Similarities of x^Q with x_k^S:

$$a(x^Q)[k] = \frac{\exp\left(\cos\left(F_\theta(x^Q), F_\theta(x_k^S)\right)\right)}{\sum_j^{N*K} \exp\left(\cos\left(F_\theta(x^Q), F_\theta(x_j^S)\right)\right)}$$

Classification probabilities for x^Q:

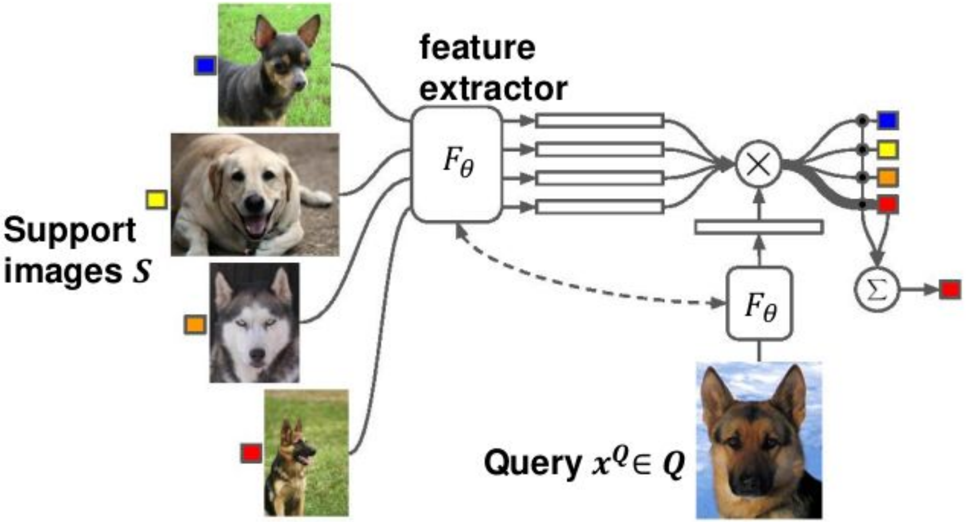
$$p = m_\varphi(x^Q) = \sum_k a(x^Q)[k] \cdot one\_hot(y_k^S)$$

"Matching networks for one shot learning", Vinyals et al. 2016

slide: Spyros Gidaris

Meta-training in Matching Networks

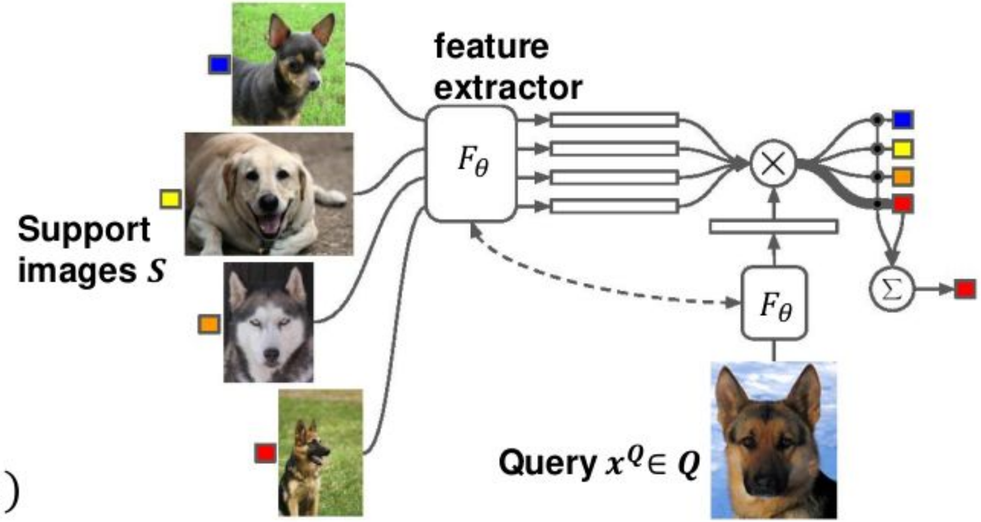
- **Meta-learner f_θ**: feature extractor $F_\theta(\cdot)$
- **Generated model m_φ**: extractor $F_\theta(\cdot)$ with support features $\{F_\theta(x_k^S),\ y_k^S\}_{k=1}^{N*K}$



"Matching networks for one shot learning", Vinyals et al. 2016

slide: Spyros Gidaris

Meta-training in Matching Networks



Meta-training routine:

1. Sample training episode (S, Q)
2. **Generate classification model** $m_\varphi = f_\theta(S) = \left\{F_\theta(\cdot),\ \left\{F_\theta\left(x_k^S\right),\ y_k^S\right\}_{k=1}^{N*K}\right\}$
3. **Predict classification scores** $p_m = m_\varphi\left(x_m^Q\right) = \sum_k a\left(x_m^Q\right)[k] \cdot one\_hot(y_k^S)$
4. Optimize θ w.r.t. the query classification loss $L(f_\theta(S), Q) = \sum_m -log(p_m[y_m^Q])$

slide: Spyros Gidaris

Matching Networks

Model	Fine Tune	5-way Acc		20-way Acc	
		1-shot	5-shot	1-shot	5-shot
BASELINE CLASSIFIER	Y	86.0%	97.6%	72.9%	92.3%
MANN (NO CONV) [21]	N	82.8%	94.9%	–	–
CONVOLUTIONAL SIAMESE NET [11]	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NET [11]	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS (OURS)	N	**98.1%**	**98.9%**	**93.8%**	98.5%
MATCHING NETS (OURS)	Y	97.9%	98.7%	93.5%	**98.7%**

Table 1: Results on the Omniglot dataset.

Model	Fine Tune	5-way Acc	
		1-shot	5-shot
BASELINE CLASSIFIER	Y	38.4%	51.2%
MATCHING NETS (OURS)	N	44.2%	57.0%
MATCHING NETS (OURS)	Y	**46.6%**	**60.0%**

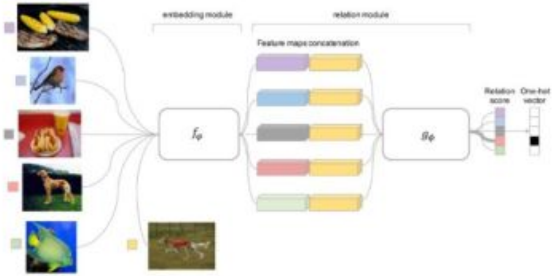
Table 2: Results on *mini*ImageNet.

- **Metric learning:**
  better results than pre-training & fine-tuning
- **Meta-training:**
  improves over siamese networks

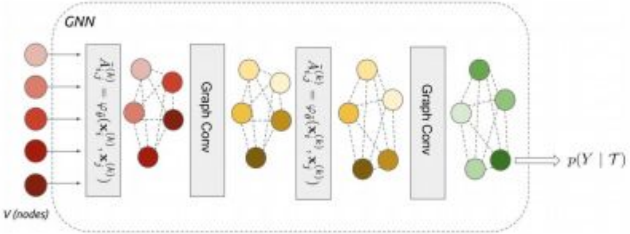
slide: Spyros Gidaris

Meta-training based metric learning

Implement distance function in prototypical nets with a relation network
"Learning to Compare: Relation Network for Few-Shot Learning", Sung et. al. 18

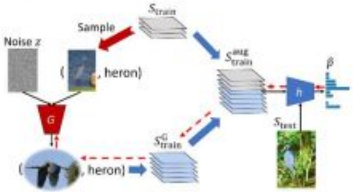


Propagate with a GNN information from the labeled support set to the query
"Few-shot Learning with Graph Neural Networks", Garcia et al. 18

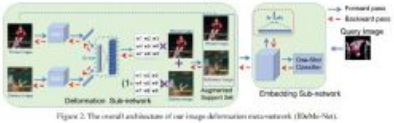


Learn to synthesize additional support examples for the metric function

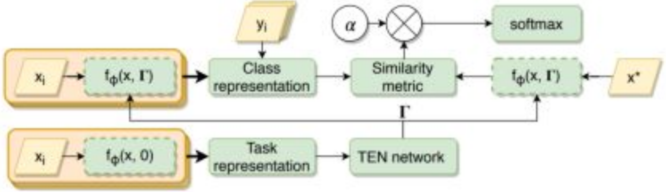
"Low-shot learning from imaginary data", Wang et.al. 18



"Image deformation meta-networks for one-shot learning", Chen et.al. 19



Task-adaptive metric function based on task-context representations
"TADAM: Task dependent adaptive metric for improved few-shot learning", Oreshkin et. al. 18



slide: Spyros Gidaris

Cosine distance based classification network

- Train typical classification network: **feature extractor + classification head**
- **Classification head:** replace dot-product (i.e., linear layer) with cosine distance

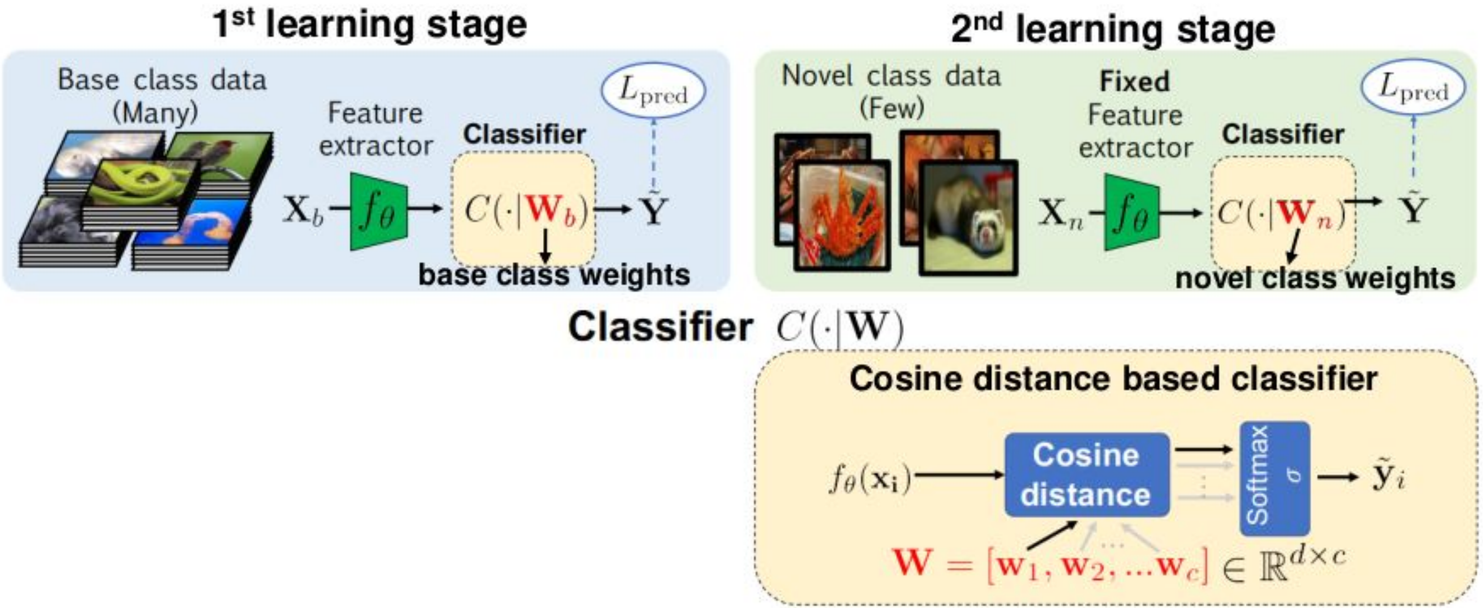


Image source (modified): W. Chen et. al. 2019

"Dynamic Few-Shot Visual Learning without Forgetting", Gidaris et al. 2018

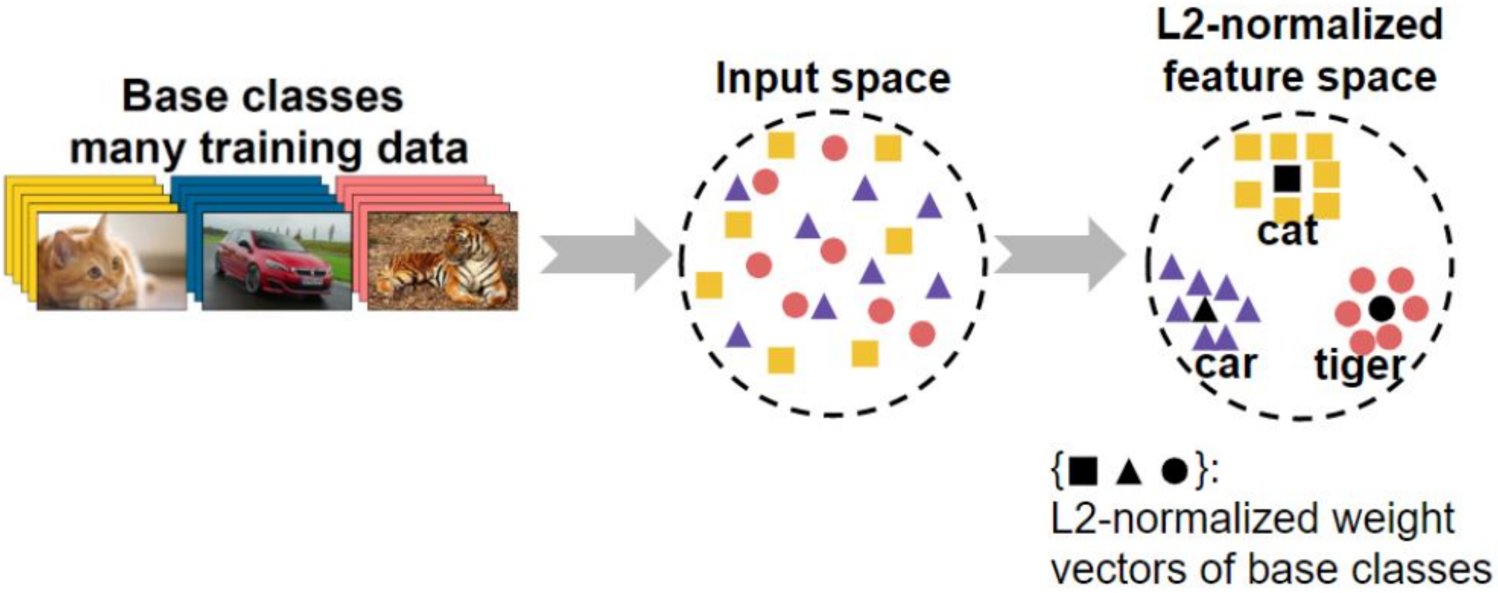
"Low-Shot Learning with Imprinted Weights", Qi et al. 2018

slide: Spyros Gidaris

Why distance based classification head?

Enforces **similar behavior as metric learning models:**

- Given an image, the learned feature must maximize (minimize) cosine similarity with weight vector of the correct class (incorrect classes)



slide: Spyros Gidaris

Cosine distance based classification network

- **1st learning stage:** standard training using the base class data
  - Trains the extractor $\boldsymbol{f_\theta}$ and classification weights $\boldsymbol{W_b}$ of base classes
  - Much simpler than meta-training based metric methods

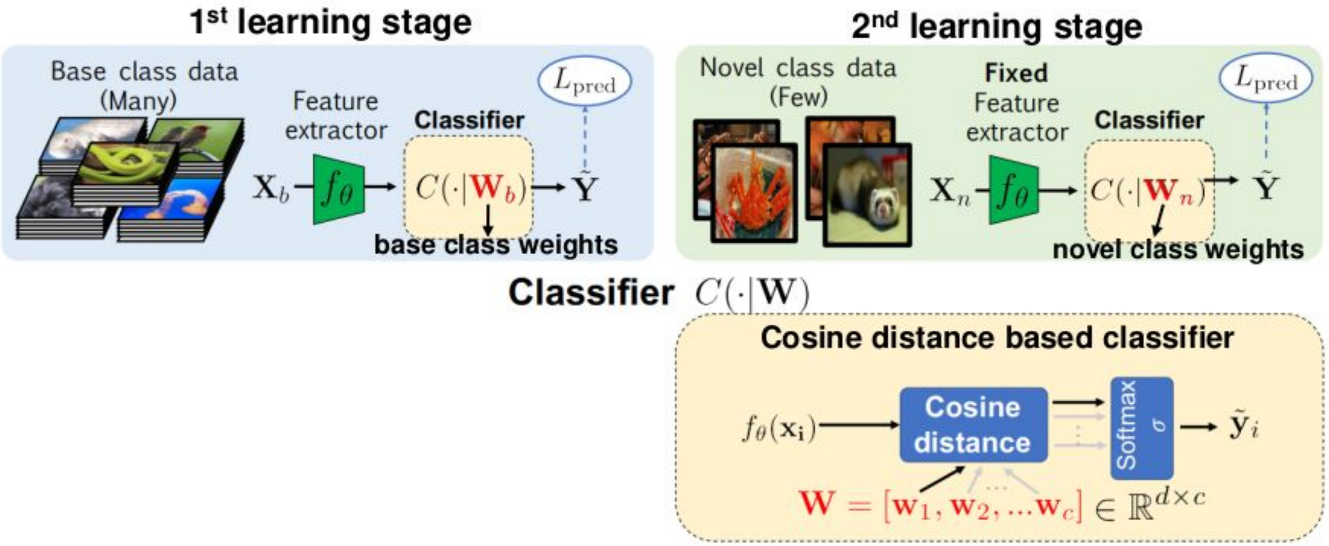


Image source (modified): Chen et al. 2019

slide: Spyros Gidaris

Cosine distance based classification network

- **2nd stage:** fix extractor $\boldsymbol{f_\theta}$ + "learn" only the classification weights W_n:
  - **compute W_n with prototypical feature averaging**

$$w_i = \frac{1}{|S_i|} \sum_{(x_k^S, y_k^S) \in S_i} f_\theta\left(x_k^S\right), \forall\, w_i \in W_n$$

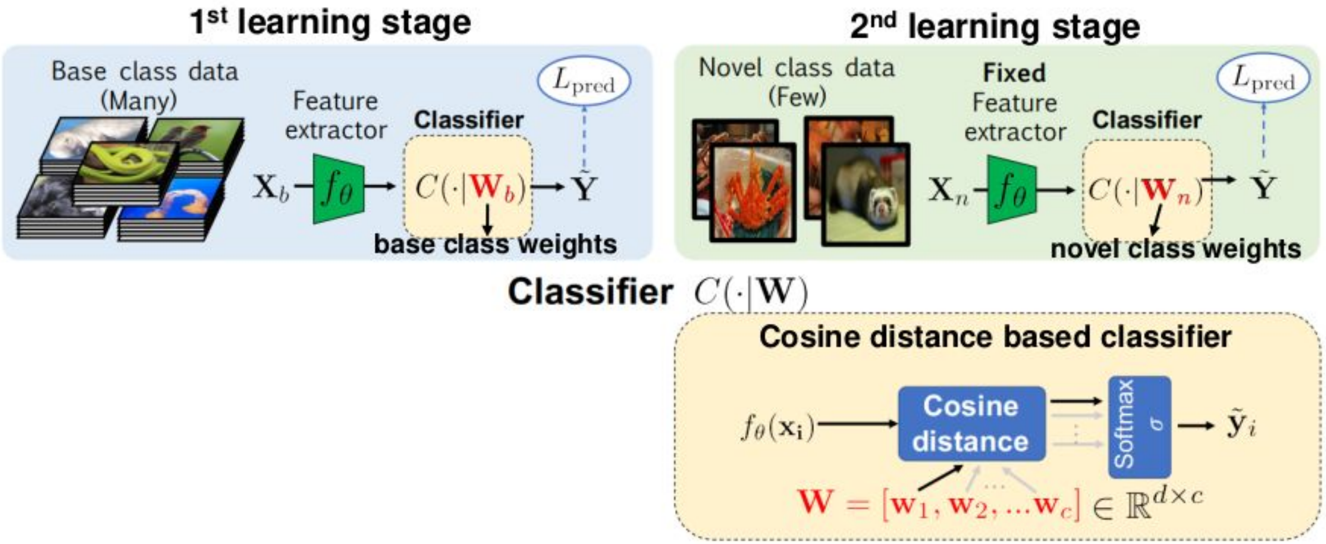


Image source (modified):
Chen et al. 2019

slide: Spyros Gidaris

Cosine classifier

Models	1-Shot	5-Shot
Matching-Nets [26]	$55.53 \pm 0.48\%$	$68.87 \pm 0.38\%$
Prototypical-Nets [23]	$54.44 \pm 0.48\%$	$72.67 \pm 0.37\%$
Cosine Classifier	$54.55 \pm 0.44\%$	$72.83 \pm 0.35\%$

Table 1: 5-way accuracies on MiniImageNet.

Simpler training with better results than Matching and Prototypical Nets

Approach	K =1	2	5	10	20
Prior work					
Prototypical-Nets	39.3	54.4	66.3	71.2	73.9
Matching Networks	43.6	54.0	66.0	72.5	76.9
Cosine Classifier	45.23	56.90	68.68	74.36	77.69

Table 2: 311-way accuracies on ImageNet-FS for *K=1, 2, 5, 10, or 20 examples per novel class.*

Source: "Dynamic Few-Shot Visual Learning without Forgetting", Gidaris et al. 18

slide: Spyros Gidaris

A Closer Look to Few-Shot Classification

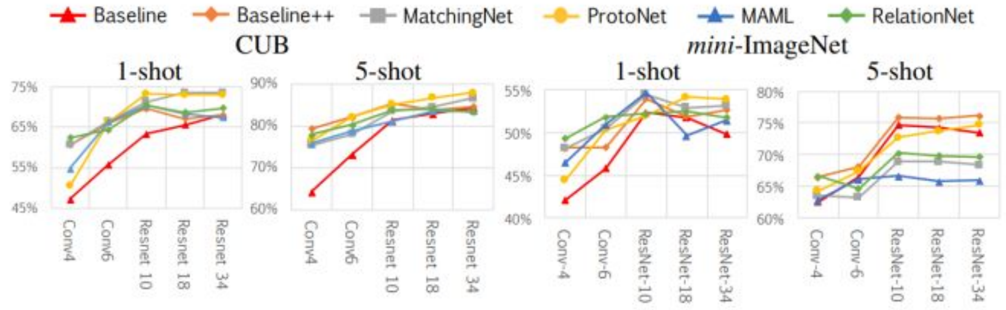


Figure 3: **Few-shot classification accuracy vs. backbone depth**. In the CUB dataset, gaps among different methods diminish as the backbone gets deeper. In *mini*-ImageNet 5-shot, some meta-learning methods are even beaten by Baseline with a deeper backbone. (Please refer to

"A Closer Look to Few-shot classification", Chen et al. 19

Method	CUB 1-shot	CUB 5-shot	*mini*-ImageNet 1-shot	*mini*-ImageNet 5-shot
Baseline	47.12 ± 0.74	64.16 ± 0.71	42.11 ± 0.71	62.53 ±0.69
Baseline++	60.53 ± 0.83	79.34 ± 0.61	48.24 ± 0.75	66.43 ±0.63
MatchingNet Vinyals et al. (2016)	60.52 ± 0.88	75.29 ± 0.75	48.14 ± 0.78	63.48 ±0.66
ProtoNet Snell et al. (2017)	50.46 ± 0.88	76.39 ± 0.64	44.42 ± 0.84	64.24 ±0.72
MAML Finn et al. (2017)	54.73 ± 0.97	75.75 ± 0.76	46.47 ± 0.82	62.71 ±0.71
RelationNet Sung et al. (2018)	62.34 ± 0.94	77.84 ± 0.68	49.31 ± 0.85	66.60 ±0.69

Baseline:
pre-training + fine-tuning last layer

Baseline++:
cosine classifier

- Meta-learning algorithms and network designs of **growing complexity**, but
- **Well-tuned baselines: often on par / better than SoTA meta-learning methods**
- **Baselines: scale better with deeper backbones**

slide: Spyros Gidaris

Recommended reading

- Deep metric learning: a (long) survey

  https://hav4ik.github.io/articles/deep-metric-learning-survey

Lab: metric learning

#4 Metric learning with triplets

- Easy few-shot learning

  https://kevinmusgrave.github.io/pytorch-metric-learning/

  - MNIST using TripletMarginLoss

    https://colab.research.google.com/github/KevinMusgrave/pytorch-metric-learning/blob/master/examples/notebooks/TripletMarginLossMNIST.ipynb

tip: you can run any github notebook in colab by replacing **`github.com`** with **`colab.research.google.com/github`**

PML

PyTorch Metric Learning

ECI 34
VIRTUAL