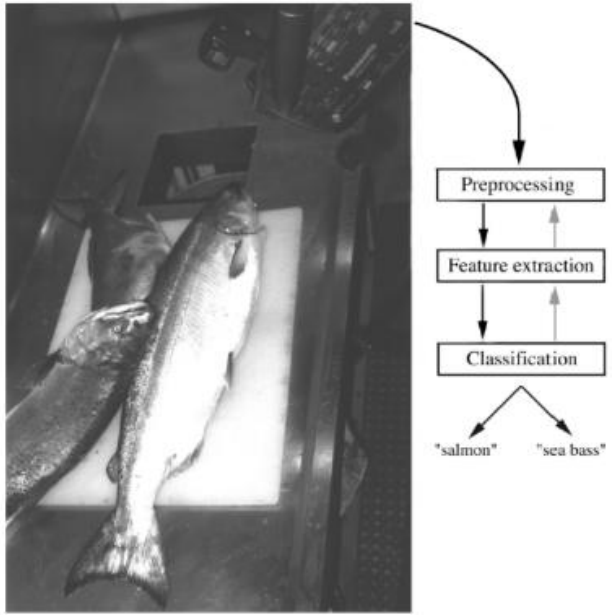
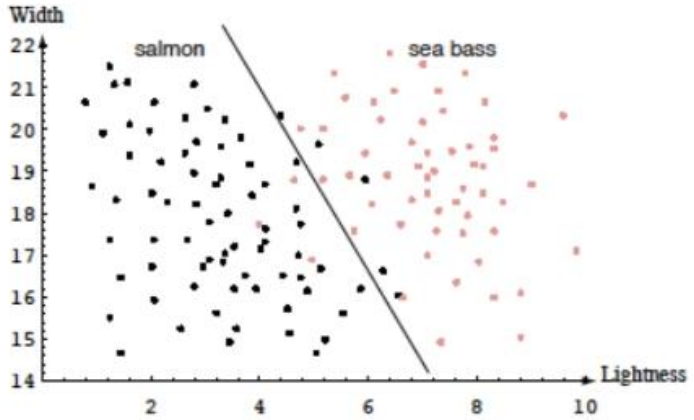
Relaxation Labelling, GAME Theory and deep learning

Ismail Elezi

Many slides are adapted from various lectures of prof. Marcello Pelillo

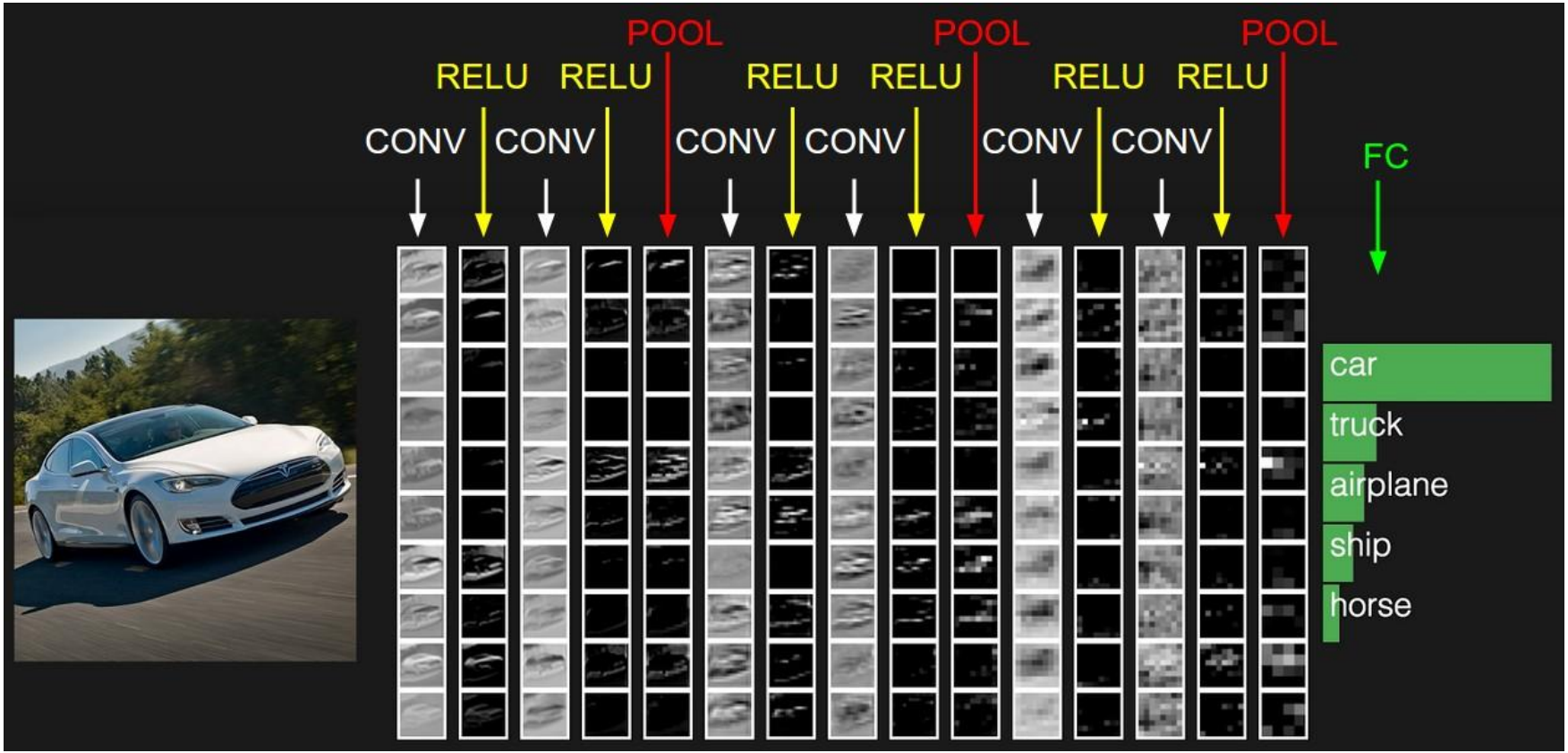
The Standard Approach



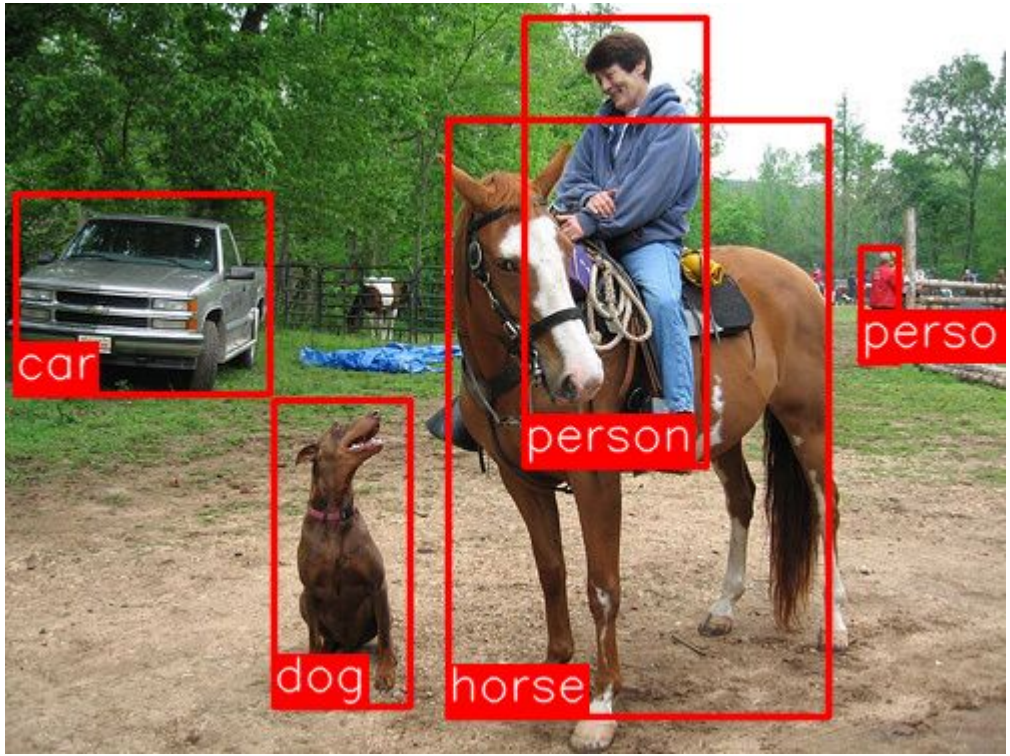


From: Duda, Hart and Stork, *Pattern Classification* (2000)

The Standard Approach



The Standard Approach



Tacit Assumptions

1. Objects possess “intrinsic” (or essential) properties.
2. Objects live in a vacuum.

On both cases, relations are neglected!

The many types of relations

- Similarity relations between objects.
- Similarity relations between categories.
- Contextual relations.
- ...

Domains: NLP, Computer Vision, Computational Biology, Medical Image Analysis, Social Network Analysis, etc.

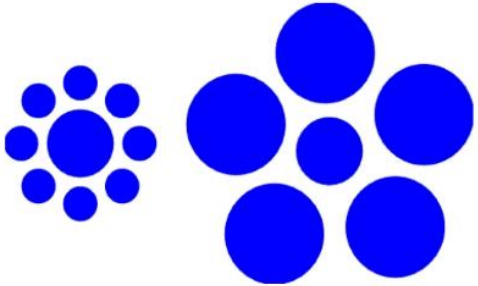
Context helps

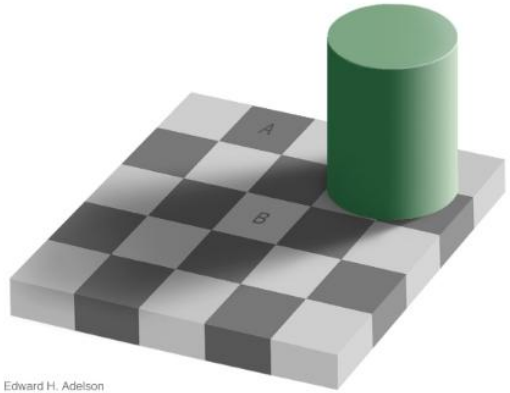
c	→	cat
	→	circus
i	→	sin
	→	fine
e	→	red
	→	read

festival

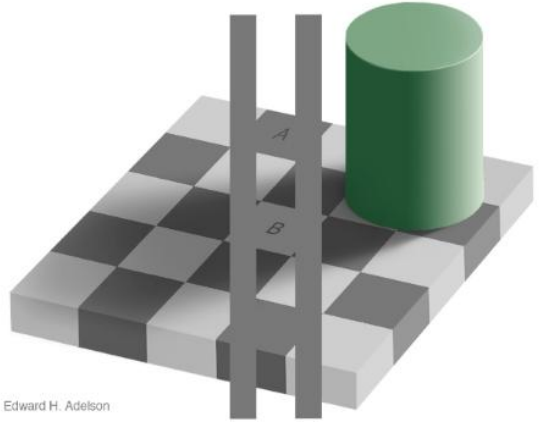
graphics

But it can also deceive





Edward H. Adelson



Edward H. Adelson

Beyond features?

The field is showing an increasing propensity towards relational approaches, e.g.,

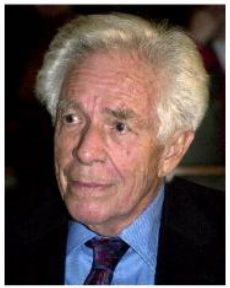
- ✓ Kernel methods
- ✓ Pairwise clustering (e.g., spectral methods, game-theoretic methods)
- ✓ Graph transduction
- ✓ Dissimilarity representations (Duin et al.)
- ✓ Theory of similarity functions (Blum, Balcan, ...)
- ✓ Relational / collective classification
- ✓ Graph mining
- ✓ Contextual object recognition
- ✓ ...

See also “link analysis” and the parallel development of “network science” ...

Why game theory?

1. Because it works.
2. Because it allows to deal with context-aware problems, non-Euclidean, non-metric, high-order and whatever you like (dis)similarities.
3. Because it allows us to go beyond convex optimizations (and many problems are non-convex).
4. Because it has finally met traditional machine learning and deep learning (GANs).

What is game theory?



"The central problem of game theory was posed by von Neumann as early as 1926 in Göttingen. It is the following:

If n players, $P_1,\ldots, P_n$, play a given game Γ, how must the i^{th} player, P_i, play to achieve the most favorable result for himself?"

Harold W. Kuhn

Lectures on the Theory of Games (1953)

A few cornerstones of game theory

1921–1928: Emile Borel and John von Neumann give the first modern formulation of a mixed strategy along with the idea of finding minimax solutions of normal-form games.

1944, 1947: John von Neumann and Oskar Morgenstern publish *Theory of Games and Economic Behavior*.

1950–1953: In four papers John Nash made seminal contributions to both non-cooperative game theory and to bargaining theory.

1972–1982: John Maynard Smith applies game theory to biological problems thereby founding "evolutionary game theory."

late 1990's –: Development of algorithmic game theory...

Prisoner's Dilemma

		Prisoner 2	
		Confess (defect)	Deny (cooperate)
Prisoner 1	Confess (defect)	-10 , -10	-1 , -25
	Deny (cooperate)	-25 , -1	-3 , -3

How to "Solve" the Game?

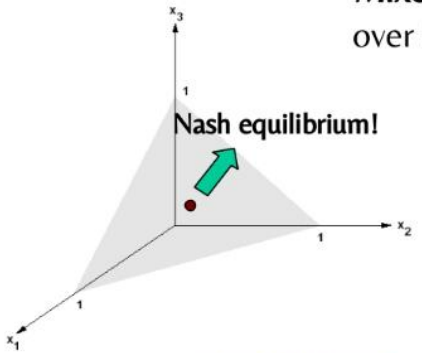
Dominated strategy !

	Prisoner 2: Confess (defect)	Prisoner 2: Deny (cooperate)
Prisoner 1: Confess (defect)	-10 , -10	-1 , -25
Prisoner 1: Deny (cooperate)	-25 , -1	-3 , -3

Dominated strategy !

Mixed strategies

Mixed strategy = probability distribution $\mathbf{x}=(x_1,\ldots,x_n)^T$ over the set of "pure" strategies

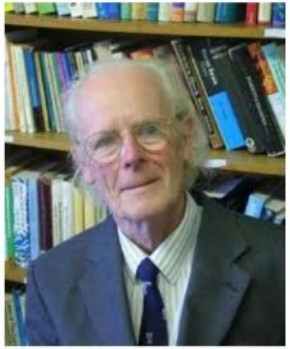


		You		
		Rock	Scissors	Paper
Me	Rock	0 , 0	1 , -1	-1 , 1
	Scissors	-1 , 1	0 , 0	1 , -1
	Paper	1 , -1	-1 , 1	0 , 0

Evolutionary game theory

«We repeat most emphatically that our theory is thoroughly static. A dynamic theory would unquestionably be more complete and therefore preferable.»

John von Neumann and Oskar Morgenstern
Theory of Games and Economic Behavior (1944)



«Paradoxically, it has turned out that game theory is more readily applied to biology than to the field of economic behaviour for which it was originally designed.»

John Maynard Smith
Evolution and the Theory of Games (1982)

Evolutionary game theory

Assumptions:

- ✓ A large population of individuals belonging to the same species which compete for a particular limited resource
- ✓ This kind of conflict is modeled as a two-player (symmetric) game, the players being pairs of randomly selected population members
- ✓ Players do not behave "rationally" but act according to a pre-programmed behavioral pattern (pure strategy)
- ✓ Utility is measured in terms of Darwinian fitness, or reproductive success

Key notion:

Evolutionary Stable Strategies (ESS's) = "stable" version of Nash equilibria.

Related to dominant-set clustering (Rota Bulò and Pelillo, 2017)

Finding ESS's and Nash equilibria: Replicator dynamics

Replicator dynamics are a populare way to find ESS's and are motivated by Darwin's principle of natural selection:

$$x_i(t+1) = x_i(t)\frac{A(x(t))_i}{x(t)^T Ax(t)}$$

where $x_i(t)$ is the population share playing strategy i at time t, and A is the payoff matrix.

MATLAB implementation

```
distance=inf;
while distance>epsilon
    old_x=x;
    x = x.*(A*x);
    x = x./sum(x);
    distance=pdist([x,old_x]');
end
```

The (Consistent) labeling problem

A **labeling problem** involves:

- ✓ A set of n **objects** $B = \{b_1,...,b_n\}$
- ✓ A set of m **labels** $\Lambda = \{1,...,m\}$

The goal is to label each object of B with a label of Λ.

To this end, two sources of information are exploited:

- ✓ Local measurements which capture the salient features of each object viewed in isolation
- ✓ Contextual information, expressed in terms of a real-valued n^2 x m^2 matrix of **compatibility coefficients** $R = \{r_{ij}(\lambda,\mu)\}$.

The coefficient $r_{ij}(\lambda,\mu)$ measures the strenght of compatibility between the two hypotheses: "b_i is labeled λ" and "b_i is labeled μ".

Relaxation Labeling Processes

The initial local measurements are assumed to provide, for each object $b_i \in B$, an m-dimensional (probability) vector:

$$p_i^{(0)} = \left(p_i^{(0)}(1), \cdots, p_i^{(0)}(m) \right)^T$$

with $p_i^{(0)}(\lambda) \geq 0$ and $\sum_\lambda \; p_i^{(0)}(\lambda) = 1$. Each $p_i^{(0)}(\lambda)$ represents the initial, non-contextual degree of confidence in the hypothesis "b_i is labeled λ".

By concatenating vectors $p_1^{(0)}, \ldots, p_n^{(0)}$ one obtains an (initial) **weighted labeling assignment** $p^{(0)} \in \mathfrak{R}^{nm}$.

The space of weighted labeling assignments is

$$IK = \underbrace{\Delta \times \ldots \times \Delta}_{m \text{ times}}$$

where each Δ is the standard simplex of $\mathfrak{R}^n$. Vertices of IK represent unambiguous labeling assignments

A relaxation labeling process takes the initial labeling assignment $p^{(0)}$ as input and iteratively updates it taking into account the compatibility model R.

Relaxation labeling processes

In a now classic 1976 paper, Rosenfeld, Hummel, and Zucker introduced the following update rule (assuming a non-negative compatibility matrix):

$$p_i^{(t+1)}(\lambda) = \frac{p_i^{(t)}(\lambda) q_i^{(t)}(\lambda)}{\sum_{\mu} p_i^{(t)}(\mu) q_i^{(t)}(\mu)}$$

where

$$q_i^{(t)}(\lambda) = \sum_j \sum_{\mu} r_{ij}(\lambda,\mu) p_i^{(t)}(\mu)$$

quantifies the support that context gives at time t to the hypothesis "b_i is labeled with label λ".

See (Pelillo, 1997) for a rigorous derivation of this rule in the context of a formal theory of consistency.

Relaxation Labeling and Polymatrix Games

As observed by Miller and Zucker (1991) the consistent labeling problem is equivalent to a polymatrix game.

Indeed, in such formulation we have:

- ✓ Objects = players
- ✓ Labels = pure strategies
- ✓ Weighted labeling assignments = mixed strategies
- ✓ Compatibility coefficients = payoffs

and:

- ✓ Consistent labeling = Nash equilibrium
- ✓ Strictly consistent labeling = strict Nash equilibrium

Further, the RHZ update rule corresponds to discrete-time multi-population "replicator dynamics" used in evolutionary game theory (see next part).

Relaxation labelling - going into details

Labels are subject to contextual constraints, expressed as an n by n block matrix.

$$R = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{n1} & \cdots & R_{nn} \end{bmatrix},$$

Each entry in the matrix represents the compatibility between:

λ on object b_i and μ on object b_j

$$R_{ij} = \begin{bmatrix} r_{ij}(1,1) & \cdots & r_{ij}(1,m) \\ \vdots & \ddots & \vdots \\ r_{ij}(m,1) & \cdots & r_{ij}(m,m) \end{bmatrix}.$$

Relaxation labelling - the update rules

P is updated (as before):

$$p_{i\lambda}^{(t+1)} = \frac{p_{i\lambda}^{(t)} q_{i\lambda}^{(t)}}{\sum_{\mu=1}^{m} p_{i\mu}^{(t)} q_{i\mu}^{(t)}},$$

Q is updated:

$$q_{i\lambda}^{(t)} = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_{j\mu}^{(t)}$$

Problems

1. Infeasible (matrix R is huge O((#objects * #labels)^2) in space. Imagine using ReLab in image classification on ImageNet, the matrix R has roughly 2e+18 entries.
2. The training set is needed to be used during inference time. K-NN in steroids?

A solution is required!

What if we use a Context window?

Context window:

$$q_{i\lambda}^{(t)} = \sum_{\delta \in \Delta_i} \sum_{\mu=1}^{m} r_{\delta\lambda\mu} p_{i+\delta,\mu}^{(t)}$$

Cost function (LS):

$$E_\gamma^{(Q)}(\boldsymbol{r}) = \frac{1}{2} \sum_{i=1}^{n_\gamma} \sum_{\lambda=1}^{m} (p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r}) - p_{i\lambda}^{(L_\gamma)})^2$$

Cost function (CE):

$$E_\gamma^{(I)}(\boldsymbol{r}) = -\sum_{i=1}^{n_\gamma} \ln p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r})$$

It’s all convex optimization now, yipes ...

On a non-convex surface … So what, ANN have been doing it always.

The Algorithm (Math, math, math)

Input: An initial feasible compatibility vector $\boldsymbol{r}^{(0)}$;
Output: An "optimal" compatibility vector.

1) $k := 0$;
2) determine the indices of active constraints, that is $J^{(k)} = \{(d, \alpha, \beta) : r^{(k)}_{d\alpha\beta} = 0\}$;
3) evaluate the vector $\boldsymbol{u}^{(k)}$, as follows:

$$u^{(k)}_{d\alpha\beta} = \begin{cases} \dfrac{\partial E(\boldsymbol{r}^{(k)})}{\partial r_{d\alpha\beta}}, & \text{if } (d, \alpha, \beta) \notin J^{(k)}, \\ 0, & \text{if } (d, \alpha, \beta) \in J^{(k)}; \end{cases}$$

4) if $u^{(k)} \neq \mathbf{0}$
   - 4.1) determine a suitable step length ρ_k;
   - 4.2) move to the next point using the relation $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \rho_k \boldsymbol{u}^{(k)}$;
   - 4.3) $k := k + 1$;
   - 4.4) goto 2);

5) else
   - 5.1) if $\partial E(\boldsymbol{r}^{(k)})/\partial r_{d\alpha\beta} \geq 0 \ \forall (d, \alpha, \beta) \in J^{(k)}$ EXIT;
   - 5.2) else
     - 5.2.1) delete from $J^{(k)}$ the index corresponding to the most negative value;
     - 5.2.2) goto 3);

Compute the derivatives - Chain rule in steroids

In the case of quadratic error function $E^{(Q)}$ we have:

$$\frac{\partial E_\gamma^{(Q)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \sum_{i=1}^{n_\gamma} \sum_{\lambda=1}^{m} (p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r}) - p_{i\lambda}^{(L_\gamma)}) \frac{\partial p_{i\lambda}^{(F_\gamma)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}}$$

while the logarithmic cost function $E^{(I)}$ yields

$$\frac{\partial E_\gamma^{(I)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = -\sum_{i=1}^{n_\gamma} \frac{\partial p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \left(p_{i\lambda_i^\gamma}^{(F_\gamma)}(\boldsymbol{r}) \right)^{-1}.$$

Compute the derivatives - Scary stuff

$$\frac{\partial p_{i\lambda}^{(t+1)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \frac{\dfrac{\partial h_{i\lambda}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \sum_{\mu=1}^{m} h_{i\mu}^{(t)}(\boldsymbol{r}) - h_{i\lambda}^{(t)}(\boldsymbol{r}) \sum_{\mu=1}^{m} \dfrac{\partial h_{i\mu}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}}}{\left(\sum_{\mu=1}^{m} h_{i\mu}^{(t)}(\boldsymbol{r}) \right)^2}$$

(

where

$$\frac{\partial h_{i\lambda}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = p_{i\lambda}^{(t)}(\boldsymbol{r}) \frac{\partial q_{i\lambda}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} + \frac{\partial p_{i\lambda}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} q_{i\lambda}^{(t)}(\boldsymbol{r}) \qquad ($$

and

$$\frac{\partial q_{i\lambda}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} = \sum_{\delta \in \Delta_i} \sum_{\mu=1}^{m} \left(\Phi(d = \delta \wedge \alpha = \lambda \wedge \beta = \mu) \cdot p_{i+\delta,\mu}^{(t)}(\boldsymbol{r}) + r_{\delta\lambda\mu} \frac{\partial p_{i+\delta,\mu}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}} \right)$$

$$= \Phi(d \in \Delta_i \wedge \alpha = \lambda) p_{i+d,\beta}^{(t)}(\boldsymbol{r}) + \sum_{\delta \in \Delta_i} \sum_{\mu=1}^{m} r_{\delta\lambda\mu} \frac{\partial p_{i+\delta,\mu}^{(t)}(\boldsymbol{r})}{\partial r_{d\alpha\beta}}.$$

It can be worse

You can actually think of more than one types of similarity (in images: up, down, right, left), (in NLP: previous word, previous 2 words, …).

Derivatives become a bit more messy.

Wait, Wait, You're selling me RNN for something else

1) In reality, it is quite similar to RNNs (with no hidden layers).
2) However, gradients are computed forward, not backward.
3) It has a Lyapunov function (it should decrease, if the implementation is correct).
4) It has better theoretical guarantees than Hopfield networks (what's that)?
5) Vanishing/Exploding gradient? Perhaps.

The Boring part is over!
Let's do some experiments

1994, Part of Speech disambiguation

TABLE I

DISAMBIGUATION ACCURACY OF RELAXATION LABELING OVER A 1,000-WORD TEST SAMPLE, USING BOTH THE INITIAL POINTS AND THE BEST POINTS FOUND BY THE LEARNING ALGORITHM

	Initial Points	Optimal Points	
		Quadratic Error	Logarithmic Error
Peleg	72.0%	88.2%	92.6%
Correlation	73.5%	93.4%	94.1%
Random	42.6%	89.7%	91.9%

Fast forward, 2.something decades later

	name	task	KB	texts	words
S7	SemEval 2007	fine grained	WN	3	444
S7CG	SemEval 2007	coarse grained	WN	5	2269
S3	Senseval 3	fine grained	WN	3	2041
S2	Senseval 2	fine grained	WN	3	2473
S13	SemEval 2013	wsd & entity disambiguation	BN	13	1931
KORE	KORE50	entity disambiguation	BN	50	146

Evaluation measure:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \cdot 100$$

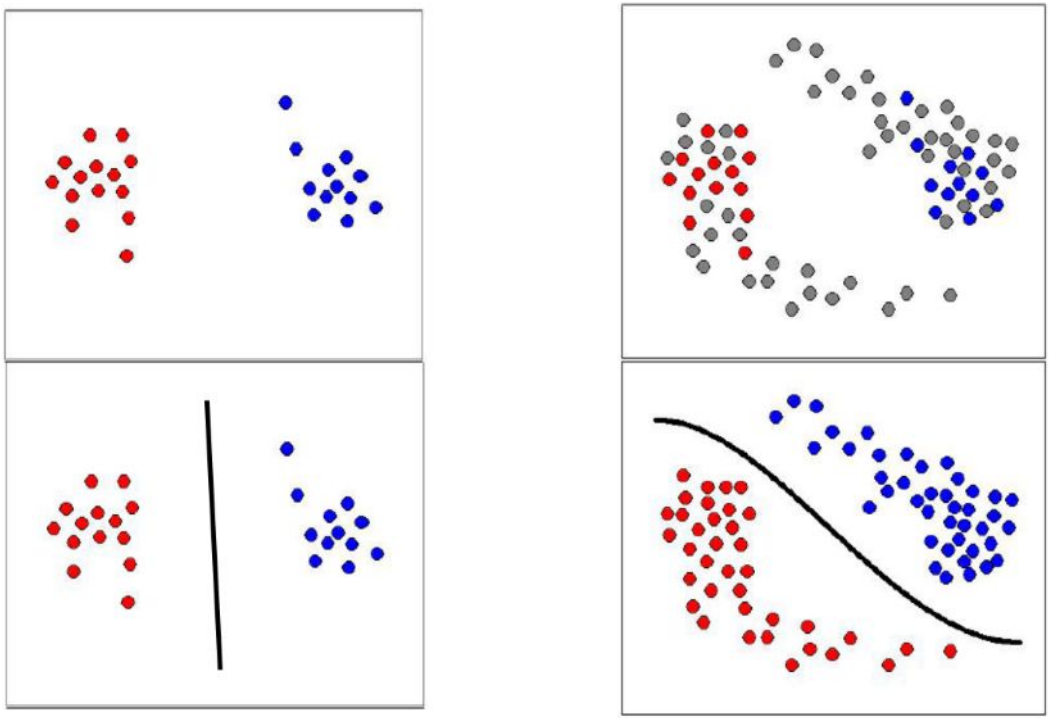
Experimental results

		S7CG	S7CG (N)	S7	S3	S2
unsup.	*Nav10*	–	–	43.1	52.9	–
	PPR_{w2w}	80.1	83.6	41.7	57.9	59.7
	WSD_{games}	**80.4***	**85.5**	**43.3**	**59.1**	**61.2**
semi sup.	*IRST-DDD-00*	–	–	–	58.3	–
	MFS	76.3	77.4	54.7	62.8	65.6*
	MRF-LP	–	–	50.6*	58.6	60.5
	Nav05	**83.2**	84.1	–	60.4	–
	PPR_{w2w}	81.4	82.1	48.6	63.0	62.6
	WSD_{games}	82.8	**85.4**	**56.5**	**64.7***	**66.0**
sup.	*Best*	82.5	82.3*	**59.1**	65.2	**68.6**
	Zhong10	**82.6**	–	58.3	**67.6**	68.2

Experimental results (entity linking)

	S13	KORE50
WSD$_{games}$	**70.8**	**75.7**
Babelfy	69.2	71.5
SUDOKU	66.3	–
MFS	66.5*	–
PPR$_{w2w}$	60.8	–
KORE	–	63.9*
GETALP	58.3	–

Application to semi-supervised learning



Adapted from: O. Duchene, J.-Y. Audibert, R. Keriven, J. Ponce, and F. Ségonne. Segmentation by transduction. *CVPR 2008.*

Graph transduction

Given a set of data points grouped into:

- ✓ labeled data: $\{(\mathbf{x}_1, y_1), \ldots, \mathbf{x}_\ell, y_\ell)\}$
- ✓ unlabeled $\{\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_n\}$ $\ell \ll n$ data:

Express data as a graph *G=(V,E)*

- ✓ *V* : nodes representing labeled and unlabeled points
- ✓ *E* : pairwise edges between nodes weighted by the similarity between the corresponding pairs of points

Goal: Propagate the information available at the labeled nodes to unlabeled ones in a "consistent" way.

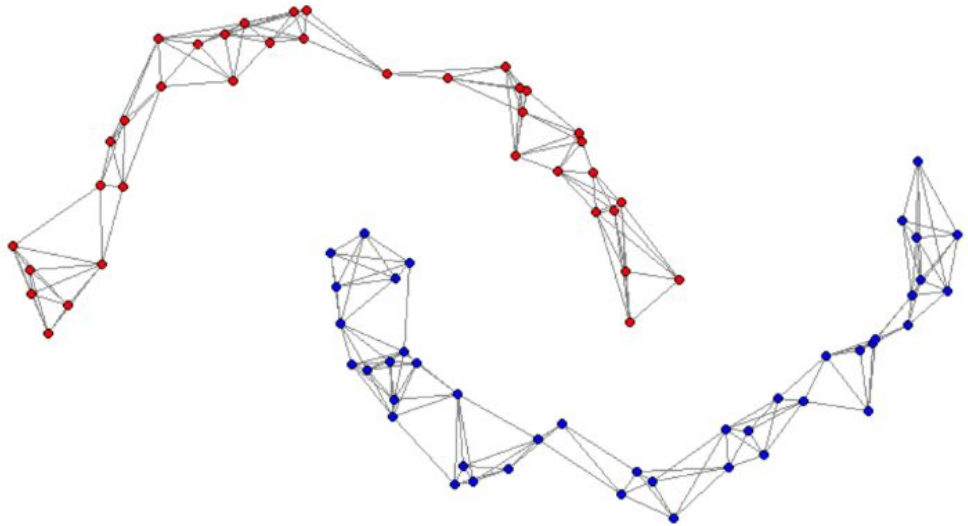
Cluster assumption:

- ✓ The data form distinct clusters
- ✓ Two points in the same cluster are expected to be in the same class

A special case

A simple case of graph transduction in which the graph G is an unweighted undirected graph:

- ✓ An edge denotes perfect similarity between points
- ✓ The adjacency matrix of G is a 0/1 matrix



The cluster assumption: Each node in a connected component of the graph should have the same class label. A constraint satisfaction problem!

The graph transduction game

Given a weighted graph G = (V, E, w), the graph trasduction game is as follow:

- ✓ Nodes = players
- ✓ Labels = pure strategies
- ✓ Weighted labeling assignments = mixed strategies
- ✓ Compatibility coefficients = payoffs

The transduction game is in fact played among the unlabeled players to choose their memberships.

- ✓ Consistent labeling = Nash equilibrium

Can be solved used standard relaxation labeling / replicator dynamics.

Applications: NLP (see next part), interactive image segmentation, content-based image retrieval, people tracking and re-identification, etc.

In short...

Graph transduction can be formulated as a non-cooperative game (i.e., a consistent labeling problem).

The proposed game-theoretic framework can cope with **symmetric, negative and asymmetric similarities** (none of the existing techniques is able to deal with all three types of similarities).

Experimental results on standard datasets show that ~~our~~ approach is not only more general but also competitive with standard approaches.

A. Erdem and M. Pelillo. Graph transduction as a noncooperative game. *Neural Computation* 24(3) (March 2012).

The “protein function predition” game

Motivation: network-based methods for the automatic prediction of protein functions can greatly benefit from exploiting *both* the similarity between proteins and the similarity between functional classes.

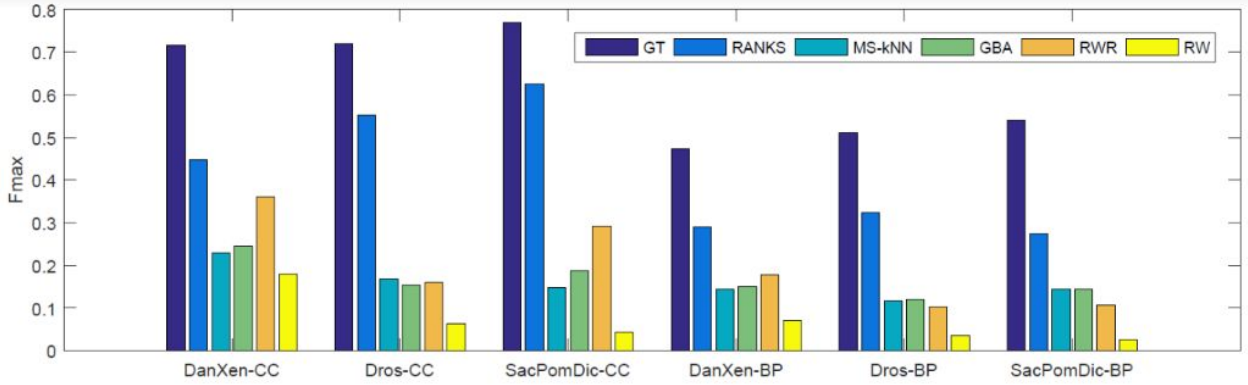
Hume’s principle: *similar* proteins should have *similar* functionalities

We envisage a (non-cooperative) game where

- Players = proteins,
- Strategies = functional classes
- Payoff function = combination of protein- and function-level similarities

Nash equilibria turn out to provide consistent functional labelings of proteins.

Preliminary results



Networks: DanXen (includes zebrafish and frog proteins), Dros (fruit fly), SacPomDic (includes the proteins of three unicellular eukaryotes).

CC = cellular component / BP = biological processs

Number of nodes (proteins): from 3195 (Dros) to 15836 (SacPomDic)

CC terms (classes): from 184 to 919

BP terms (classes): from 2281 to 5037

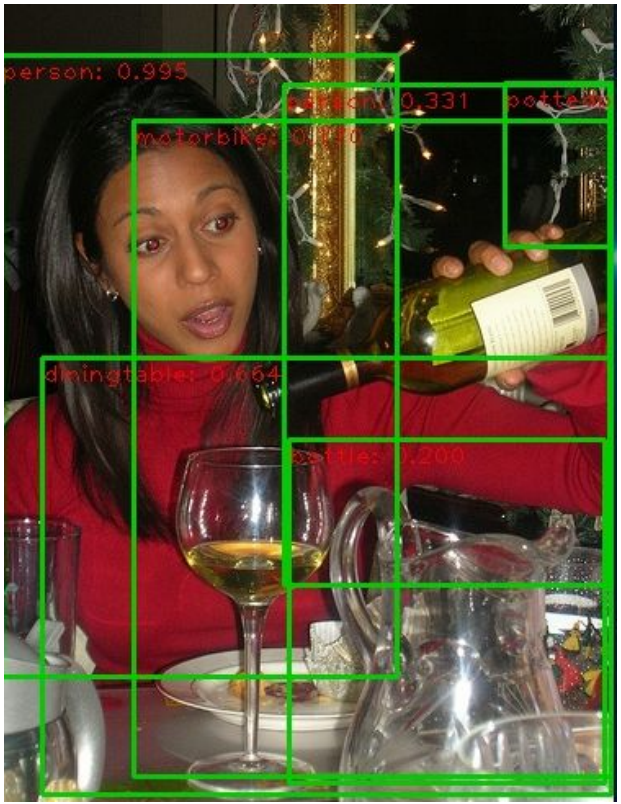
Competitors

- Random Walk (RW)
- Random Walk with Restart (RWR)
- Funckenstein (GBA)
- Multi Source-kNN method (MS-kNN)
- RANKS

My goal is to combine relab with deep learning (starting from R-CNN)

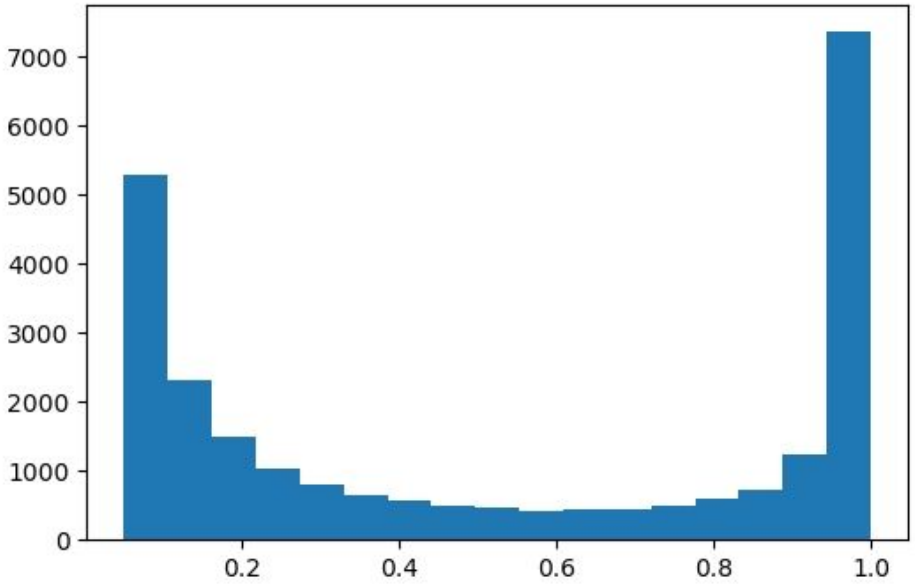
Step 1 - Train a faster R-CNN

Web has code that can do it.



Step 1 - Train a faster R-CNN

There is a lot of room for improvement on an R-CNN (VOC Pascal MAP is circa 0.7, MS-COCO MAP is circa 0.4).



Step 2 - Train Relab

We have code about it (needs to be GPU-ized):

```
def dpdr(self, p, q, grad_p, grad_q):
    h = p * q
    grad_h = p * grad_q + grad_p * q  # eq 23

    grad_p = (grad_h * h.sum(axis=1)[:, np.newaxis] - h * grad_h.sum(axis=1)[:, np.newaxis]) / (
        (h.sum(axis=1)[:, np.newaxis]) ** 2.0)

    grad_p[np.isnan(grad_p)] = 0.0
    return grad_p

def dqdr(self, p, grad_p, N, d, alpha, beta):
    grad_q = np.tensordot(np.tensordot(self.R, grad_p.T, axes=(2, 0)), N, axes=([0, 2], [0, 2])).T
    grad_q[:, alpha] += (N[d, :, :] * p[np.newaxis, :, beta]).sum(axis=1)
    return grad_q
```

Step 3 - Combine

1. For each image, get the soft labels (vector of probabilities) and assign it to p.

2. Update p, using the compatibilities we learned in step (2).

3. Profit?

Acknowledgments




BIBLIOGRAPHY

1) R. A. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. IEEE tPAMI (1983).
2) M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. IEEE Trans. IEEE tPAMI (1994)
3) M. Pelillo. The dynamics of nonlinear relaxation labeling processes. Journal of Mathematical Imaging and Vision (1997)
4) A. Erdem and M. Pelillo. Graph Transduction as a Non-Cooperative Game. Neural Computation (2012)
5) R. Tripodi and M. Pelillo. A game theoretic approach to word sense disambiguation. Computational Linguistics (2017)

Thank You!

QUESTIONS???