

# Multi-Target Prediction: A Unifying View on Problems and Methods

Willem Waegeman

Research Unit Knowledge-based Systems (KERMIT)  
Department of Mathematical Modelling, Statistics and Bioinformatics  
Ghent University, Belgium

October 13 2017

# Oranje verslaat Zweden, maar gaat niet naar WK

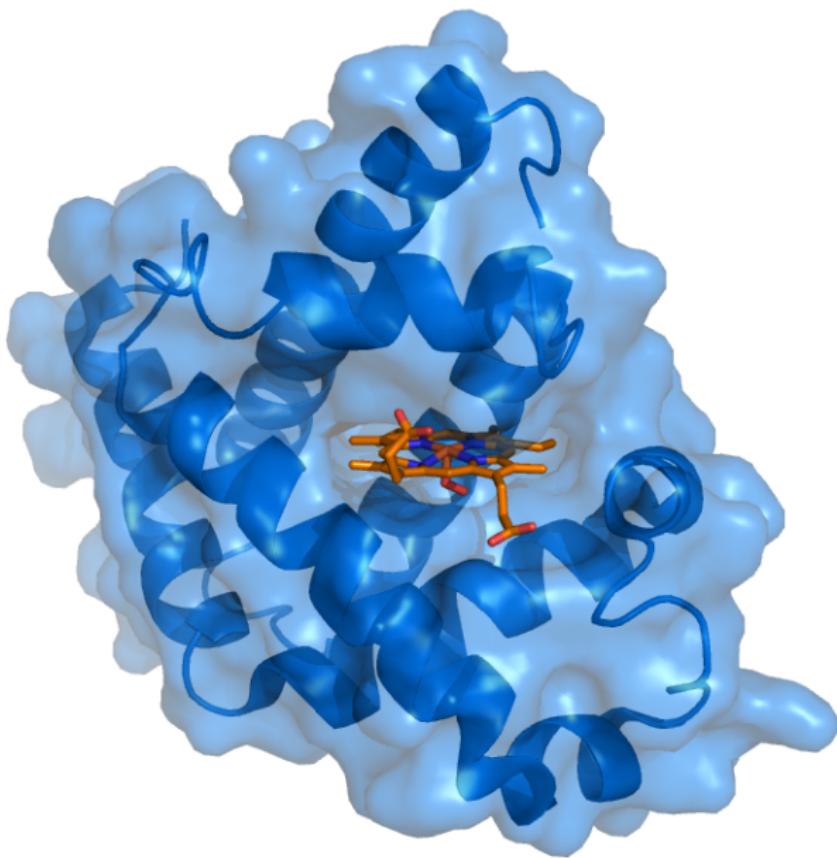
© DINSDAG, 22:38 VOETBAL

Het Nederlands elftal heeft zich zoals verwacht niet geplaatst voor de play-offs om de laatste vier Europese WK-tickets. Oranje won in Amsterdam met 2-0 van Zweden, maar het verschil had zeven of meer doelpunten moeten zijn.



## Multi-label classification: the example of document categorization

		Tennis	Football	Biking	Movies	TV	Belgium
01101	Text1	0	1	0	0	1	1
00111	Text2	1	0	0	0	0	1
01110	Text3	0	0	0	1	1	0
10001	Text4	0	0	1	0	1	0
01011	Text5	1	0	0	1	0	0
11110	Text6	?	?	?	?	?	?



# Multivariate regression: the example of protein-ligand interaction prediction

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	
01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5
11110		?	?	?	?	?	?



# Multi-task learning: the example of predicting student marks

	School1	School2	School3
01101		7	
00111		9	
01110		5	
10001		8	
01011			9
11110		?	?

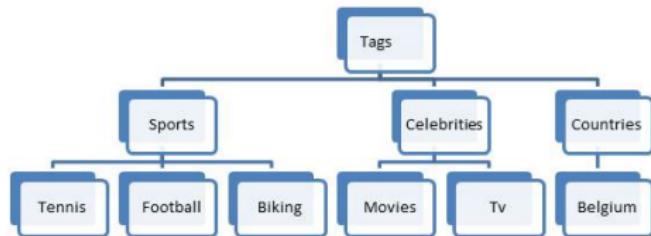
There are a lot of multi-target prediction problems around...



# Overview of this talk

- 1 A unifying view on MTP problems
- 2 A unifying view on MTP methods
- 3 Loss functions in multi-target prediction
- 4 Conclusions

Let's assume a document hierarchy:  
How would you call this machine learning problem?



		Tennis	Football	Biking	Movies	Tv	Belgium
01101	Text1	0	0	0	0	0	1
00111	Text2	0	0	1	0	1	1
01110	Text3	0	0	0	1	1	0
10001	Text4	0	0	1	0	1	0
01011	Text5	1	0	0	1	0	0

11110	Text6	?	?	?	?	?	?
-------	-------	---	---	---	---	---	---

Let's assume a target representation:  
How would you call this machine learning problem?

	0011	1100	0110
	School1	School2	School3
01101		7	
00111		9	
01110			5
10001			8
01011			9
11110		?	?

Let's assume a target representation:

How would you call this machine learning problem?



01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5

11110		?	?	?	?	?	?
-------	--	---	---	---	---	---	---

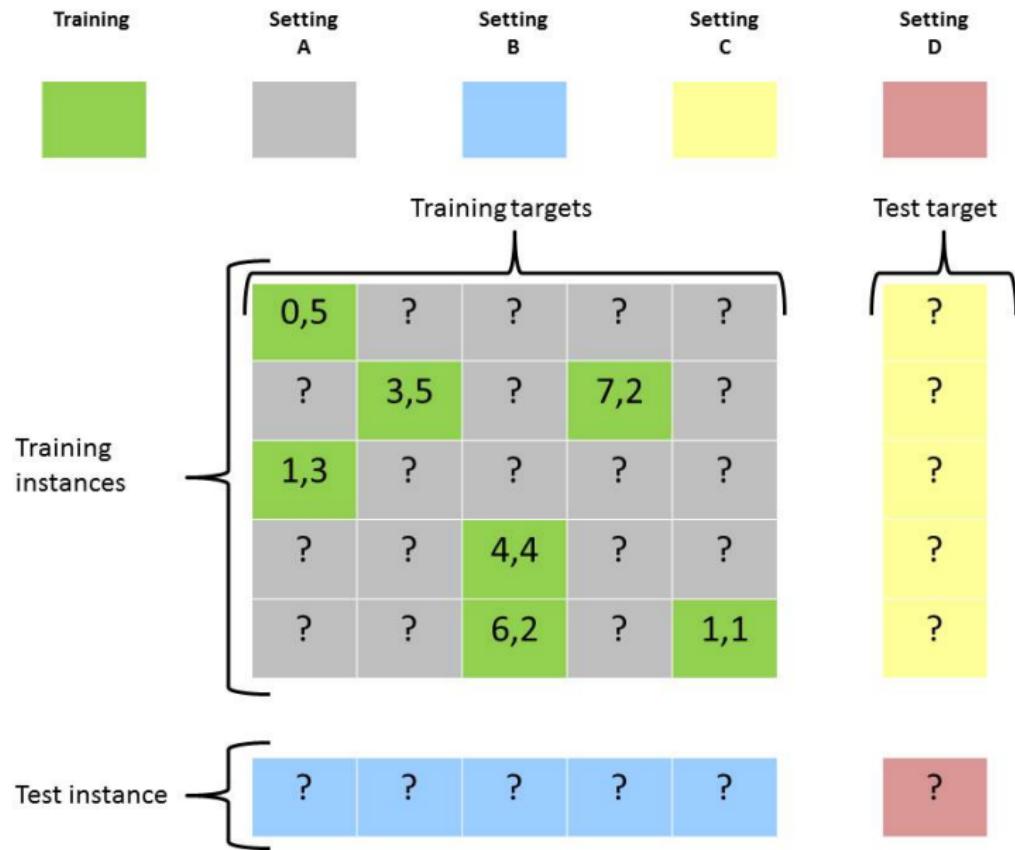
## Generalizing to new targets

$g(., .)$  : target similarity



01101		1,3	0,2	1,4	1,7	3,5	1,3	?
00111		2	1,7	1,5	7,5	8,2	7,6	?
01110		0,2	0	0,3	0,4	1,2	2,2	?
10001		3,1	1,1	1,3	1,1	1,7	5,2	?
01011		4,7	2,1	2,5	1,5	2,3	8,5	?
11110		?	?	?	?	?	?	?

# Important subdivision of different learning settings



# General framework

## Definition

A multi-target prediction setting is characterized by instances  $\mathbf{x} \in \mathcal{X}$  and targets  $\mathbf{t} \in \mathcal{T}$  with the following properties:

1. A training dataset consists of triplets  $(\mathbf{x}_i, \mathbf{t}_j, y_{ij})$ , where  $y_{ij} \in \mathcal{Y}$ .
2. In total  $n$  instances and  $m$  targets are observed during training, with  $n$  and  $m$  finite numbers.
3. As such, the scores  $y_{ij}$  of the training data can be arranged in an  $n \times m$  matrix  $Y$ .
4. The score set  $\mathcal{Y}$  is one-dimensional. It consists of nominal, ordinal or real values.
5. The goal consists of making predictions for any instance-target couple  $(\mathbf{x}, \mathbf{t}) \in \mathcal{X} \times \mathcal{T}$ .

# Overview of this talk

- 1 A unifying view on MTP problems
- 2 A unifying view on MTP methods
- 3 Loss functions in multi-target prediction
- 4 Conclusions

# A unifying view on MTP methods



Group of methods	Applicable setting
<b>Independent models</b>	B and C
Similarity-enforcing methods	B and C
Relation-exploiting methods	B, C and D
Relation-constructing methods	B and C
Representation-exploiting methods	B, C and D
Representation-constructing methods	A, B and C

A baseline method:  
learning a model for each target independently

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	
01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5
11110		?	?	?	?	?	

## A baseline method: learning a model for each target independently

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	
01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5
11110		?	?	?	?	?	?

## A baseline: Independent Models

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	
01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5
11110		?	?	?	?	?	?

## A baseline: Independent Models

Linear basis function model for  $i$ -th target:

$$f_i(\mathbf{x}) = \mathbf{a}_i^\top \phi(\mathbf{x}),$$

Solving as a joint optimization problem:

$$\min_A \|Y - XA\|_F^2 + \sum_{i=1}^m \lambda_i \|\mathbf{a}_i\|^2,$$

With the following notations:

$$X = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_n)^T \end{bmatrix} \quad A = [\mathbf{a}_1 \quad \cdots \quad \mathbf{a}_m].$$

Hamming loss as alternative for binary labels:

$$L_{Ham}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^m I(y_j = \hat{y}_j)$$

Learning a model for each target independently is still state-of-the-art in extreme multi-label classification<sup>1</sup>:

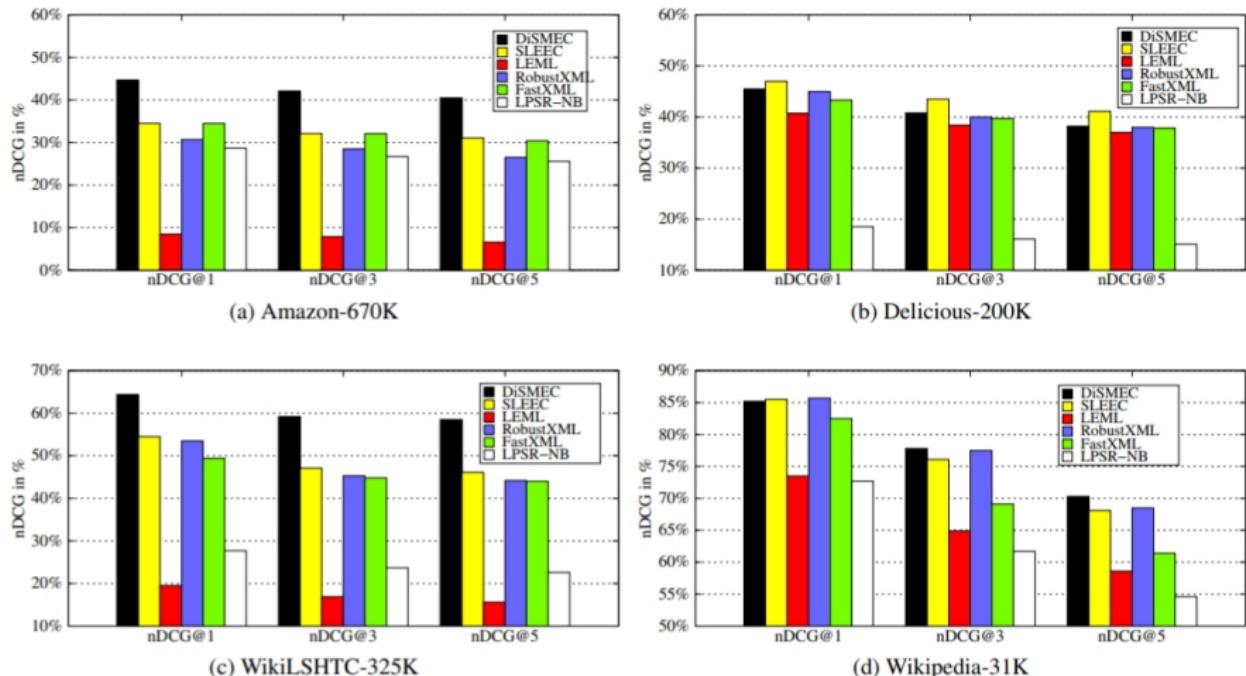


Figure 3: nDCG@k for k=1, 3 and 5

<sup>1</sup> Babbar and Schölkopf, DISMEC: Distributed Sparse Machines for Extreme Multi-label classification, WSDM 2017

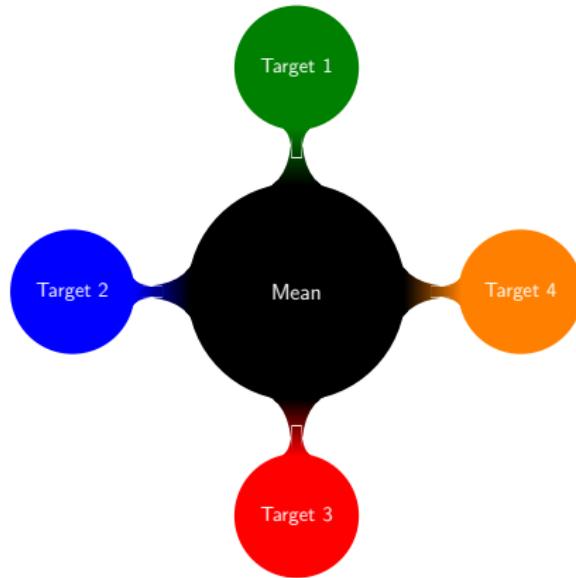
# A unifying view on MTP methods



Group of methods	Applicable setting
Independent models	B and C
<b>Similarity-enforcing methods</b>	B and C
Relation-exploiting methods	B, C and D
Relation-constructing methods	B and C
Representation-exploiting methods	B, C and D
Representation-constructing methods	A, B and C

# Mean-regularized multi-task learning<sup>2</sup>

- **Simple assumption:** models for different targets are related to each other.
- **Simple solution:** the parameters of these models should have similar values.
- **Approach:** bias the parameter vectors towards their mean vector.



$$\min_A \|Y - XA\|_F^2 + \lambda \sum_{i=1}^m \left\| \mathbf{a}_i - \frac{1}{m} \sum_{j=1}^m \mathbf{a}_j \right\|^2,$$

<sup>2</sup> Evgeniou and Pontil, Regularized multi-task learning, KDD 2004.

## Joint feature selection

- Enforce that the same features are selected for different targets<sup>3</sup>:

$$\min_A ||Y - XA||_F^2 + \lambda \sum_{j=1}^p ||\mathbf{a}_j||^2$$

- The vectors  $\mathbf{a}_j$  now represent the rows of matrix  $A^T$ :

L1-norm per target:

$$A^T = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mp} \end{bmatrix} \in \diamond$$

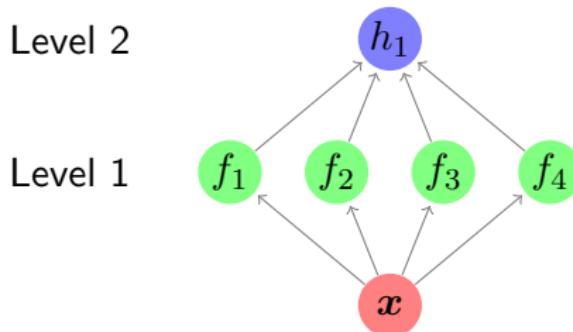
Mix L1/L2-norm per target:

$$A^T = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mp} \end{bmatrix} \in \diamond$$
$$(\mathbf{a}_1, \dots, \mathbf{a}_p) \in \diamond$$

<sup>3</sup> Obozinski et al. Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing 2010

## Stacking (Stacked generalization)

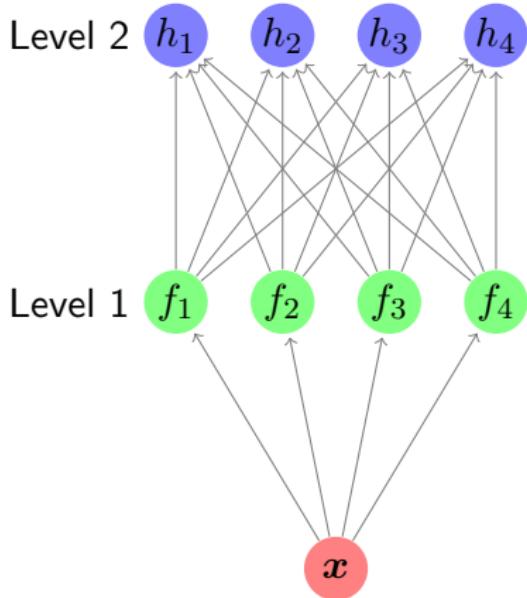
- Originally introduced as a general ensemble learning or blending technique.<sup>4</sup>
- Level 1 classifiers: apply a series of ML methods on the same dataset (or, one ML method on bootstrap samples of the dataset)
- Level 2 classifier: apply an ML method to a new dataset consisting of the predictions obtaining at Level 1



<sup>4</sup> Wolpert, Stacked generalization. Neural Networks 1992

# Stacking applied to multi-target prediction<sup>5</sup>

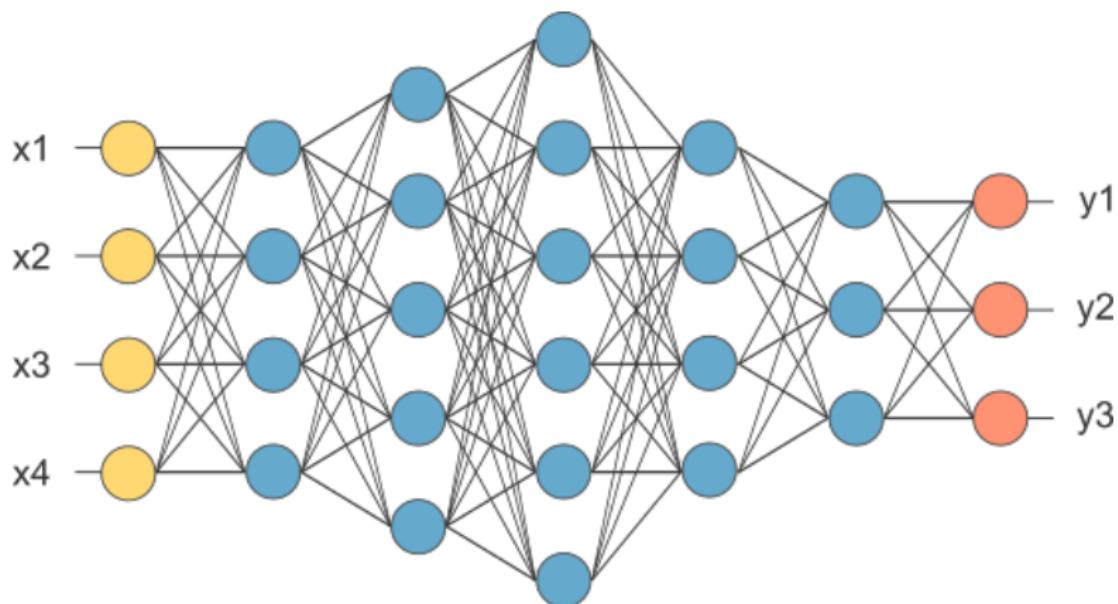
- Level 1 classifiers: learn a model for every target independently
- Level 2 classifier: learn again a model for every target independently, using the predictions of the first step as features



<sup>5</sup> Cheng and Hüllermeier, Combining Instance-based learning and Logistic Regression for Multi-Label classification, Machine Learning, 2009

# MTP in (Deep) Neural Networks

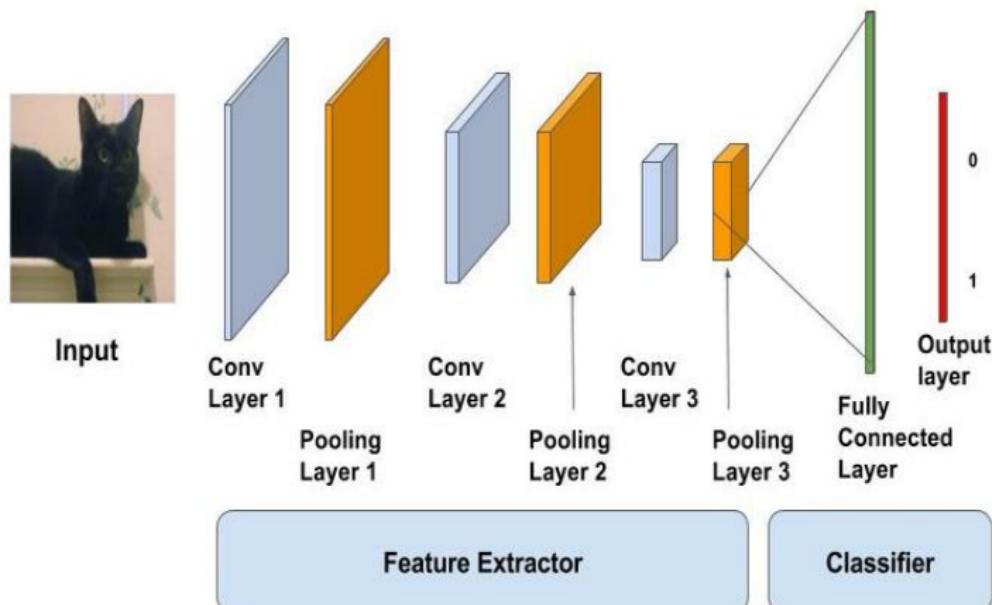
Commonly-used architecture: weight sharing among targets<sup>6</sup>



<sup>6</sup> Caruana, Multitask learning: A knowledge-based source of inductive bias. Machine Learning 1997

# Re-using Pretrained Models in (Deep) Neural Networks

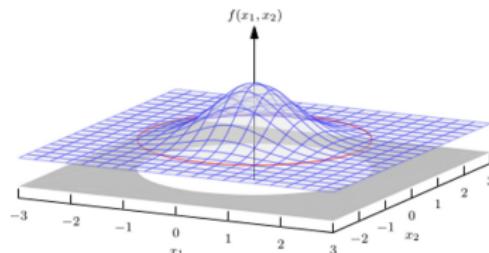
Commonly-used training method: first train on targets that have a lot of observations, only train some parameters for targets that have few observations<sup>7</sup>



<sup>7</sup> Keras Tutorial: Transfer Learning using pre-trained models

## An intuitive explanation: James-Stein estimation

- Consider a multivariate normal distribution  $\mathbf{y} \sim N(\boldsymbol{\theta}, \sigma^2 \mathbf{I})$ .

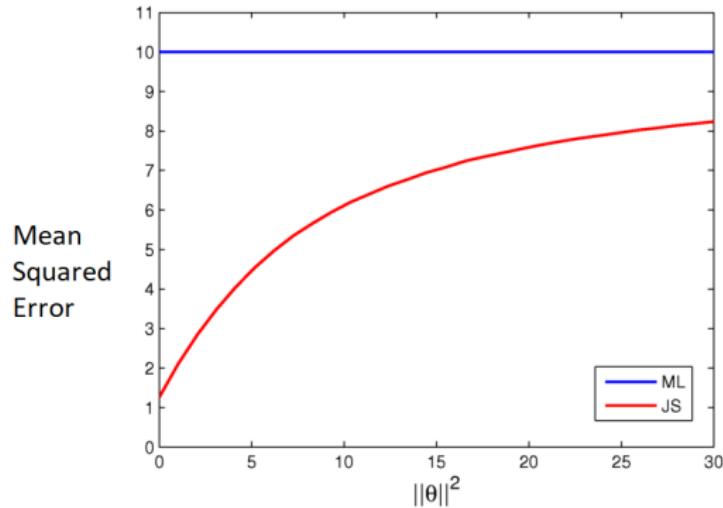


- What is the best estimator of the mean vector  $\boldsymbol{\theta}$ ?
- Evaluation w.r.t. MSE:  $\mathbb{E}[(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^2]$
- Single-observation maximum likelihood estimator:  $\hat{\boldsymbol{\theta}}^{\text{ML}} = \mathbf{y}$
- James-Stein estimator<sup>8</sup>:

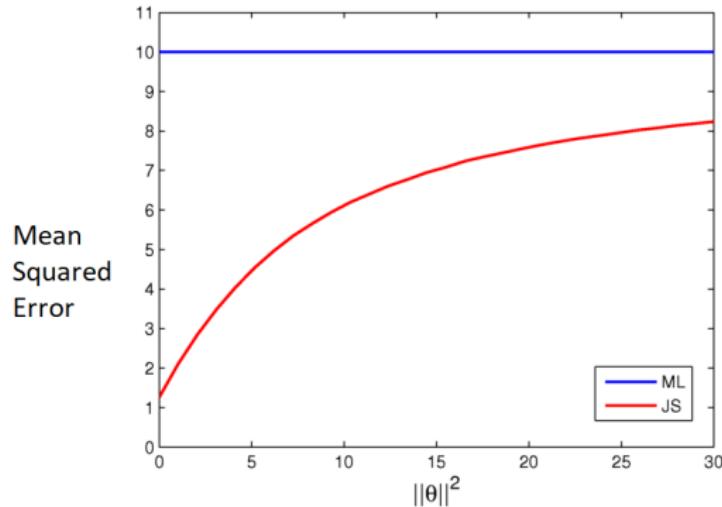
$$\hat{\boldsymbol{\theta}}^{\text{JS}} = \left( 1 - \frac{(m-2)\sigma^2}{\|\mathbf{y}\|^2} \right) \mathbf{y}$$

<sup>8</sup> W. James and C. Stein. Estimation with quadratic loss. In Proc. Fourth Berkeley Symp. Math. Statist. Prob. 1, pages 361-379, 1961

- Works best when the norm of the mean vector is close to zero:



- Works best when the norm of the mean vector is close to zero:



- Regularization towards other directions is also possible:

$$\hat{\theta}^{JS+} = \left(1 - \frac{(m-2)\sigma^2}{\|\mathbf{y} - \mathbf{v}\|^2}\right) (\mathbf{y} - \mathbf{v}) + \mathbf{v}$$

- Only outperforms the maximum likelihood estimator w.r.t. the sum of squared errors over all components.

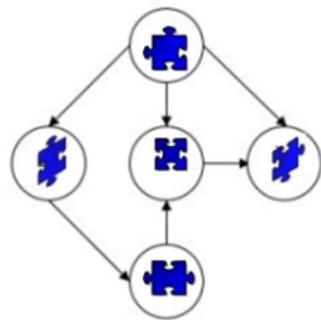
# A unifying view on MTP methods



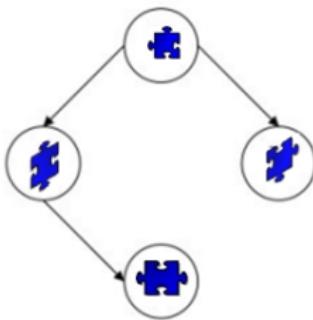
Group of methods	Applicable setting
Independent models	B and C
Similarity-enforcing methods	B and C
<b>Relation-exploiting methods</b>	B, C and D
Relation-constructing methods	B and C
Representation-exploiting methods	B, C and D
Representation-constructing methods	A, B and C

## Exploiting relations in regularization terms

Graph



Tree



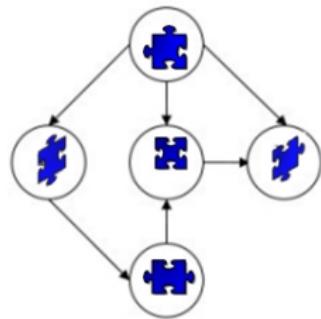
Similarity

	1	2	3	4
1	1	0.26	0.26	0.04
2	0.26	1	0.7	0.57
3	0.26	0.7	1	0.44
4	0.04	0.57	0.44	1

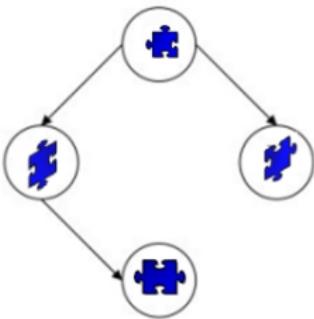
<sup>9</sup> Gopal and Yang, Recursive regularization for large-scale classification with hierarchical and graphical dependencies, KDD 2013

## Exploiting relations in regularization terms

Graph



Tree



Similarity

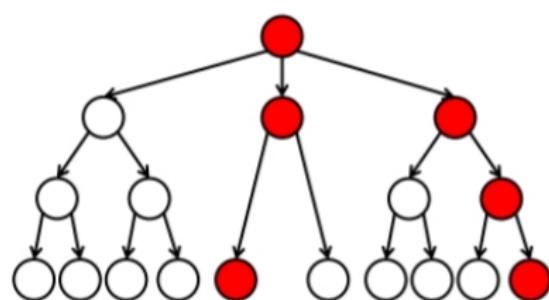
		1	2	3	4
1	1	0.26	0.26	0.04	
2	0.26	1	0.7	0.57	
3	0.26	0.7	1	0.44	
4	0.04	0.57	0.44	1	

Graph-based regularization is an approach that can be applied to the three types of relations<sup>9</sup>:

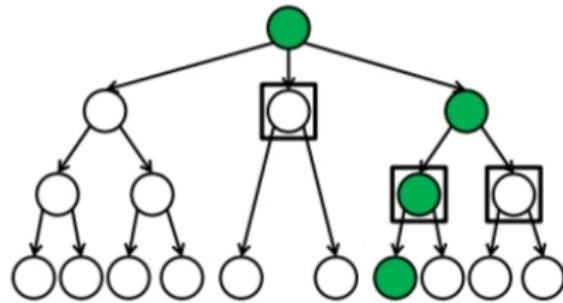
$$\min_A ||Y - XA||_F^2 + \lambda \sum_{i=1}^m \sum_{j \in \mathcal{N}(i)} ||\mathbf{a}_i - \mathbf{a}_j||^2$$

<sup>9</sup> Gopal and Yang, Recursive regularization for large-scale classification with hierarchical and graphical dependencies, KDD 2013

## Hierarchical multi-label classification



(a) Ground truth.



(b) Prediction A.

In addition to performance gains in general, hierarchies can also be used to define specific loss functions, such as the H-loss<sup>10</sup>:

$$L_H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i: y_i \neq \hat{y}_i} c_i I(anc(y_i) = anc(\hat{y}_i))$$

$c_i$  depends on the depth of node  $i$

<sup>10</sup> Bi and Kwok, Bayes-optimal hierarchical multi-label classification, IEEE Transactions on Knowledge and Data Engineering, 2014

## Exploiting similarity measures among targets

	1	0.26	0.26	0.04	
	0.26	1	0.7	0.57	
	0.26	0.7	1	0.44	
	0.04	0.57	0.44	1	

Can be done within the framework of vector-valued kernel functions<sup>11</sup>:

$$f(\mathbf{x}, \mathbf{t}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{t}) = \sum_{(\bar{\mathbf{x}}, \bar{\mathbf{t}}) \in \mathcal{D}} \alpha_{(\bar{\mathbf{x}}, \bar{\mathbf{t}})} \Gamma((\mathbf{x}, \mathbf{t}), (\bar{\mathbf{x}}, \bar{\mathbf{t}}))$$

Model the joint kernel as a product of an instance kernel  $k(\cdot, \cdot)$  and a target kernel  $g(\cdot, \cdot)$ :

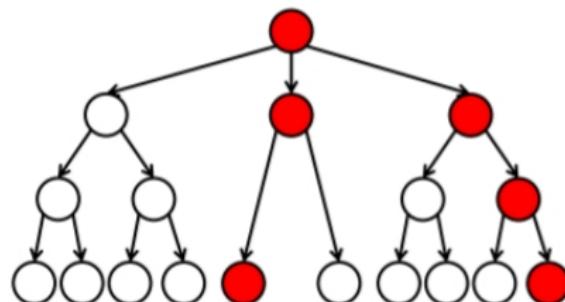
$$\Gamma((\mathbf{x}, \mathbf{t}), (\bar{\mathbf{x}}, \bar{\mathbf{t}})) = k(\mathbf{x}, \bar{\mathbf{x}}) \cdot g(\mathbf{t}, \bar{\mathbf{t}})$$

---

<sup>11</sup> Alvarez et al., Kernels for vector-valued functions: a review, Foundation and Trends in Machine Learning

# Converting graphs to similarities or target representations

- **Similarities:** use graph structure to express target similarities
  - e.g. the shortest-path kernel between two nodes
- **Representations:** often characteristics of a specific vertex or edge
  - e.g. the number of positive labels that are siblings of a vertex<sup>12</sup>



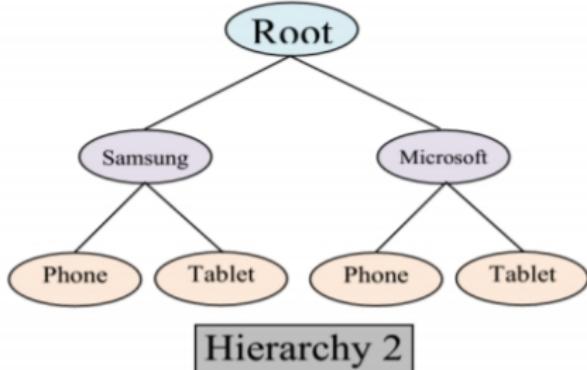
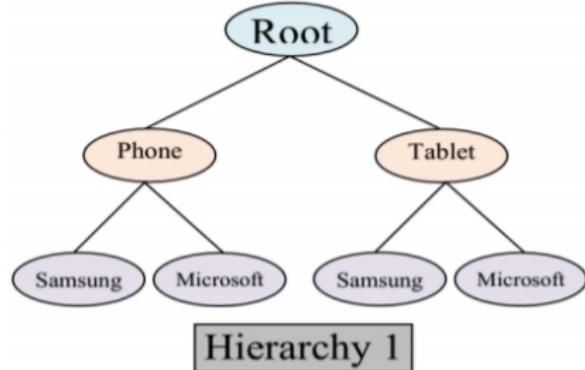
<sup>12</sup> Rousu et al., Kernel-based learning of hierarchical multilabel classification models, JMLR 2006

# A unifying view on MTP methods



Group of methods	Applicable setting
Independent models	B and C
Similarity-enforcing methods	B and C
Relation-exploiting methods	B, C and D
<b>Relation-constructing methods</b>	B and C
Representation-exploiting methods	B, C and D
Representation-constructing methods	A, B and C

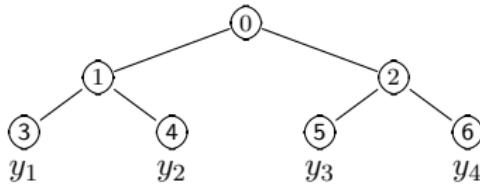
# Constructing target hierarchies



- It might be difficult for a human expert to define a hierarchy<sup>13</sup>
- Perhaps one can try to learn the hierarchy from data?
- Algorithms: level flattening, node removal, hierarchy modification, hierarchy generation, etc.

<sup>13</sup> Rangwala and Naik, Tutorial on Large-Scale Hierarchical Classification, KDD 2017.

## Label trees ( $\neq$ decision trees)



- Organize classifiers in a tree structure (one leaf  $\Leftrightarrow$  one label)
- Mainly used in multi-class and multi-label classification
- Goal is fast prediction: almost logarithmic in the number of labels
- Algorithms: Label embedding trees<sup>14</sup>, Nested dichotomies<sup>15</sup>, Conditional probability trees<sup>16</sup>, Hierarchical softmax<sup>17</sup>, FastText<sup>18</sup>, Probabilistic classifier chains<sup>19</sup>

---

<sup>14</sup> Bengio et al., Label embedding trees for large multi-class tasks, NIPS 2010

<sup>15</sup> Frank and Kramer, Ensembles of nested dichotomies for multi-class problems, ICML 2004

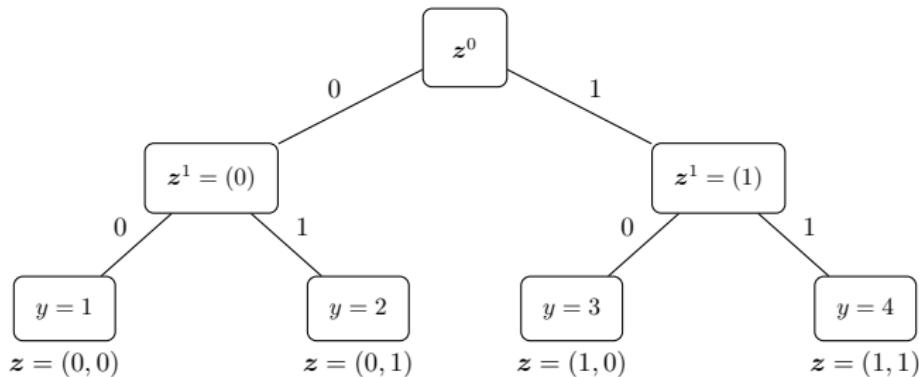
<sup>16</sup> Beygelzimer et al., Conditional probability tree estimation analysis and algorithms. UAI 2009

<sup>17</sup> Morin and Bengio, Hierarchical probabilistic neural network language model, AISTATS 2005

<sup>18</sup> Joulin et al., Bag of tricks for efficient text classification. CoRR, abs/1607.01759, 2016

<sup>19</sup> Dembczynski et al., Bayes optimal multilabel classification via probabilistic classifier chains, ICML 2010

# Hierarchical softmax / Probabilistic classifier trees



- Encode the targets by a **prefix code** ( $\Rightarrow$  tree structure)<sup>20</sup>
- Multi-class classification: each label  $y$  **coded** by  $z = (z_1, \dots, z_l) \in \mathcal{C}$
- Multi-label classification: a label vector  $y = (y_1, \dots, y_m)$  is a prefix code.

<sup>20</sup> Dembczynski et al., Consistency of probabilistic classifier trees. ECMLPKDD 2016

## Probabilistic classifier chains

- Estimate the joint conditional distribution  $P(\mathbf{Y} \mid \mathbf{x})$ .
- For optimizing the subset 0/1 loss:

$$\ell_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = \llbracket \mathbf{y} \neq \hat{\mathbf{y}} \rrbracket$$

- Repeatedly apply the **product rule of probability**:

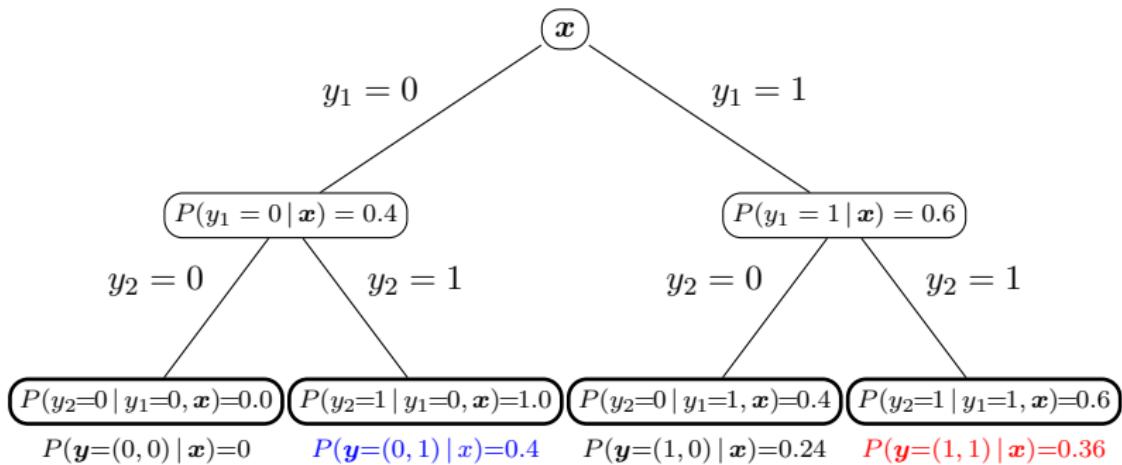
$$P(\mathbf{Y} = \mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^m P(Y_i = y_i \mid \mathbf{x}, y_1, \dots, y_{i-1}).$$

- Learning relies on constructing probabilistic classifiers for estimating

$$P(Y_i = y_i \mid \mathbf{x}, y_1, \dots, y_{i-1}),$$

independently for each  $i = 1, \dots, m$ .

- Inference relies on exploiting a probability tree:



- For subset 0/1 loss one needs to find  $\mathbf{h}(\mathbf{x}) = \arg \max_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} \mid \mathbf{x})$ .
- Greedy and approximate search techniques with guarantees exist.<sup>21</sup>
- Other losses: compute the prediction on a sample from  $P(\mathbf{Y} \mid \mathbf{x})$ .<sup>22</sup>

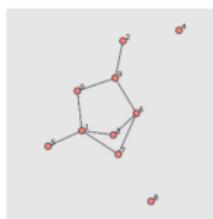
<sup>21</sup> Kumar et al., Beam search algorithms for multilabel learning, Machine Learning 2013

<sup>22</sup> Dembczynski et al., An analysis of chaining in multi-label classification, ECAI 2012

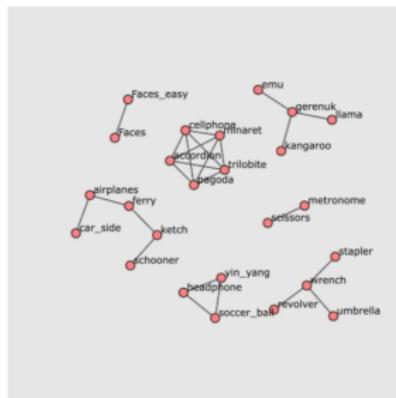
# Constructing target similarities by output kernel learning

- Consider models  $f : \mathcal{X} \rightarrow \mathbb{R}^m$
- Training dataset  $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$
- Learnable correlation matrix  $\Gamma$  between targets
- Learn output kernel and model parameters jointly<sup>23</sup>:

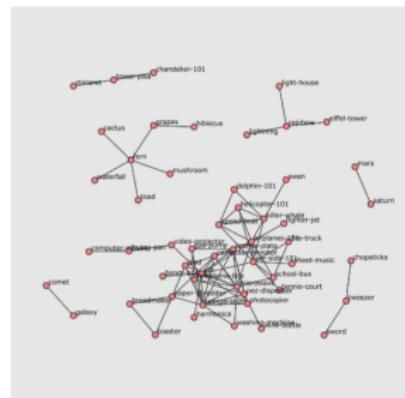
$$\min_{\Gamma \in \mathbb{R}^{m \times m}} \left[ \min_{f \in \mathcal{F}} \sum_{i=1}^n \frac{\|f(\mathbf{x}_i) - \mathbf{y}_i\|_2^2}{2\lambda} + \frac{\|f\|_{\mathcal{F}}^2}{2} + \frac{\|\Gamma\|_{\mathcal{F}}^2}{2} \right]$$



(a) USPS digits



(b) Caltech 101

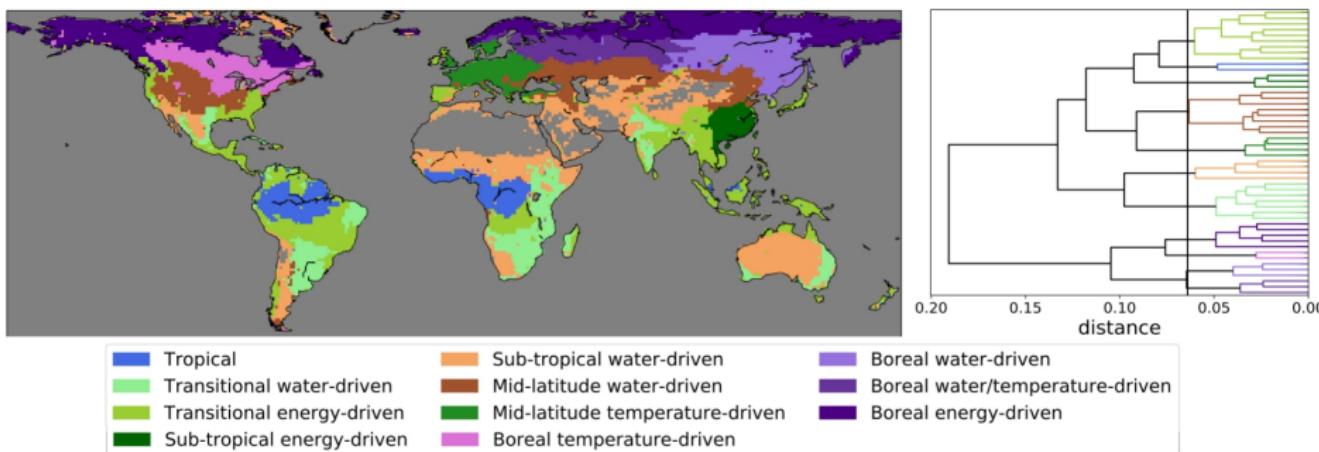


(c) Caltech 256

<sup>23</sup> Dinuzzo et al., Learning Output Kernels with Block Coordinate Descent, ICML 2011

# Constructing hierarchies to obtain additional insight

- Application in climate science
- Result of learning 20000 tasks simultaneously with a multi-task learning method
- Followed by hierarchical clustering of the learned weight vectors<sup>24</sup>:



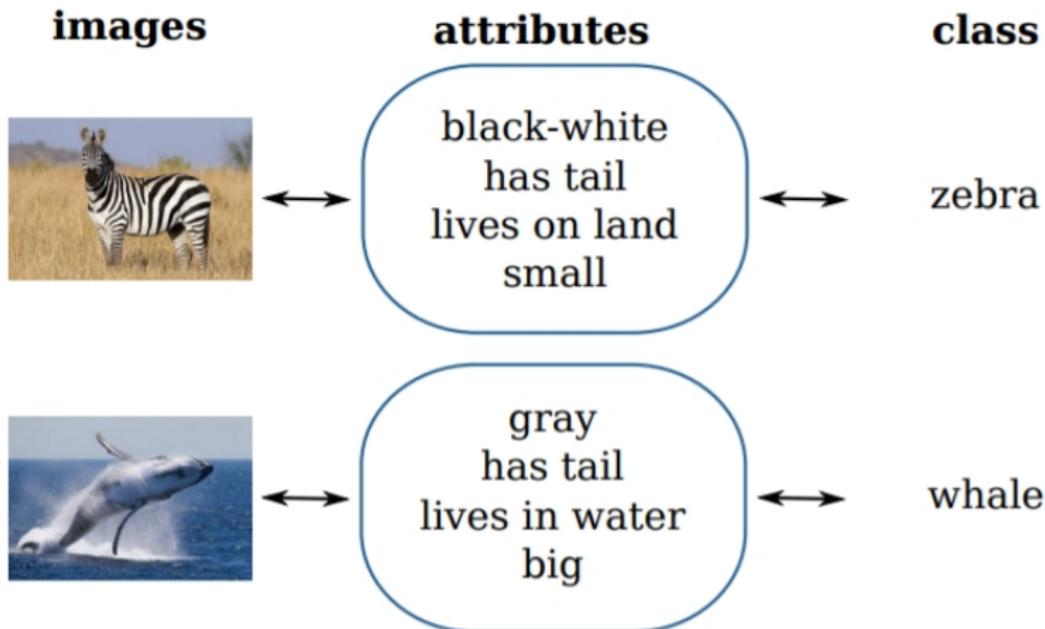
<sup>24</sup> Papagiannopoulou et al. Global hydro-climatic biomes identified with multi-task learning, Geoscientific Model Development Discussions 2018

# A unifying view on MTP methods



Group of methods	Applicable setting
Independent models	B and C
Similarity-enforcing methods	B and C
Relation-exploiting methods	B, C and D
Relation-constructing methods	B and C
<b>Representation-exploiting methods</b>	B, C and D
Representation-constructing methods	A, B and C

# A target representation in computer vision



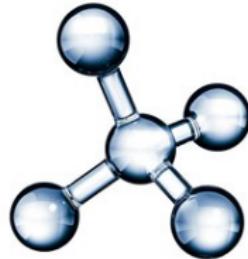
Target representations are the key element of zero-shot learning methods<sup>25</sup>

<sup>25</sup> Examples taken from the CVPR 2016 Tutorial on Zero-shot learning for Computer Vision

# Target representations can take many forms



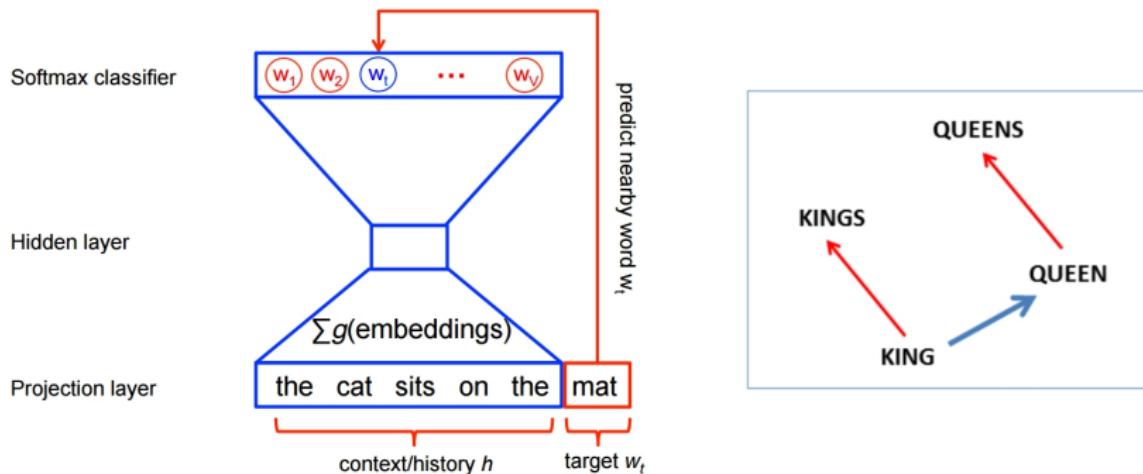
A T G G C T A C  
T A C G G A T A



# Learning target embeddings from text: Word2Vec

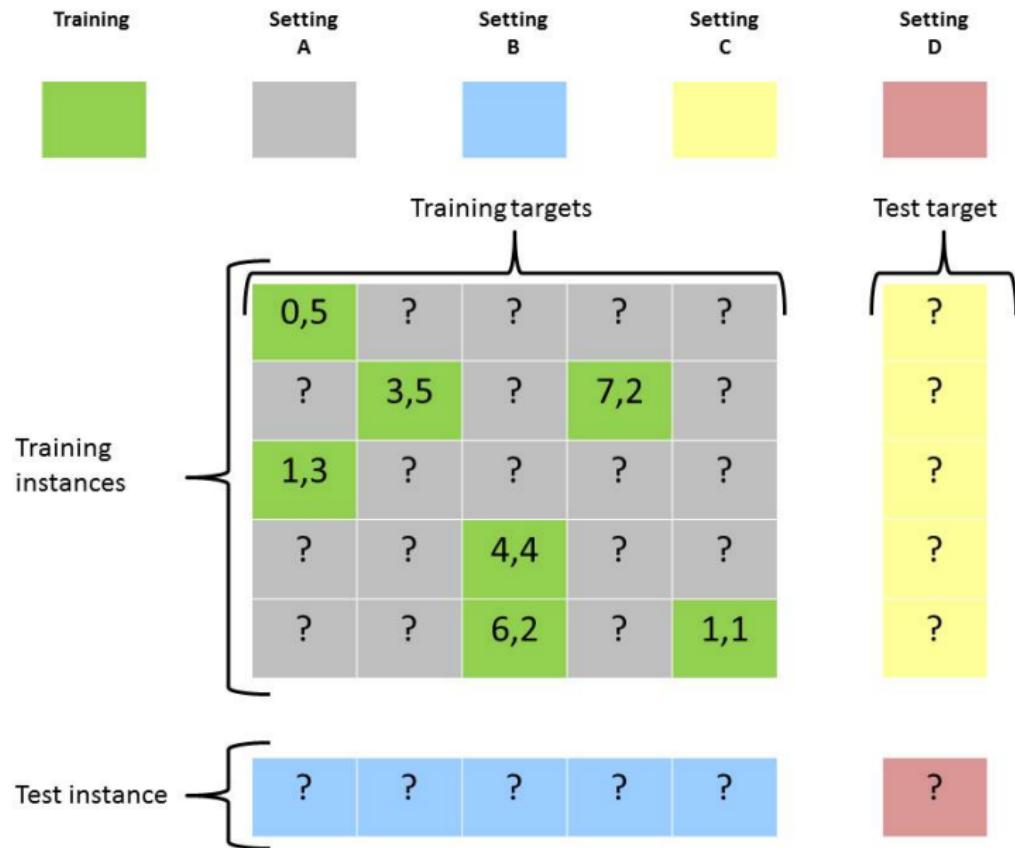
Predict the probability of the next word  $w_t$  given the previous words  $h^{26}$ :

$$P(w_t | h) = \frac{\exp(f(w_t, h))}{\sum_{\text{allwords}} \exp(f(w_t, h))}$$



<sup>26</sup> Mikolov et al., Efficient Estimation of Word Representations in Vector Space, Arxiv 2013

# Different learning settings revisited



# Kronecker kernel ridge regression

Pairwise model representation in the primal:

$$f(\mathbf{x}, \mathbf{t}) = \mathbf{w}^T (\phi(\mathbf{x}) \otimes \psi(\mathbf{t}))$$

Kronecker product pairwise kernel in the dual<sup>27</sup>:

$$f(\mathbf{x}, \mathbf{t}) = \sum_{(\bar{\mathbf{x}}, \bar{\mathbf{t}}) \in \mathcal{D}} \alpha_{(\bar{\mathbf{x}}, \bar{\mathbf{t}})} k(\mathbf{x}, \bar{\mathbf{x}}) \cdot g(\mathbf{t}, \bar{\mathbf{t}}) = \sum_{(\bar{\mathbf{x}}, \bar{\mathbf{t}}) \in \mathcal{D}} \alpha_{(\bar{\mathbf{x}}, \bar{\mathbf{t}})} \Gamma((\mathbf{x}, \mathbf{t}), (\bar{\mathbf{x}}, \bar{\mathbf{t}}))$$

Least-squares minimization with  $\mathbf{z} = \text{vec}(Y)$ :

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\Gamma}\boldsymbol{\alpha} - \mathbf{z}\|_2^2 + \lambda \boldsymbol{\alpha}^\top \boldsymbol{\Gamma} \boldsymbol{\alpha}$$

---

<sup>27</sup> Stock et al., A comparative study of pairwise learning methods based on kernel ridge regression, Neural Computation 2018

# Two-step zero-shot learning<sup>28 29</sup>

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	Mol7
01101	1,3	0,2	1,4	1,7	3,5	1,3	?
00111	2	1,7	1,5	7,5	8,2	7,6	?
01110	0,2	0	0,3	0,4	1,2	2,2	?
10001	3,1	1,1	1,3	1,1	1,7	5,2	?
01011	4,7	2,1	2,5	1,5	2,3	8,5	?
11110	?	?	?	?	?	?	?

<sup>28</sup> Pahikkala et al. A two-step approach for solving full and almost full cold-start problems in dyadic prediction, ECML/PKDD 2014.

<sup>29</sup> Romero-Paredes and Torr, An embarrassingly simple approach to zero-shot learning, ICML 2015.

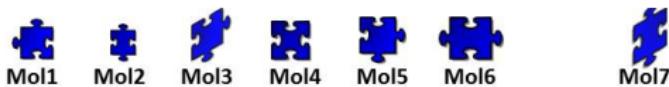
# Two-step zero-shot learning<sup>30 31</sup>

	Mol1	Mol2	Mol3	Mol4	Mol5	Mol6	
01101		1,3	0,2	1,4	1,7	3,5	1,3
00111		2	1,7	1,5	7,5	8,2	7,6
01110		0,2	0	0,3	0,4	1,2	2,2
10001		3,1	1,1	1,3	1,1	1,7	5,2
01011		4,7	2,1	2,5	1,5	2,3	8,5
11110		1,2	2,1	1,7	4,3	2,4	2,5

<sup>30</sup> Pahikkala et al. A two-step approach for solving full and almost full cold-start problems in dyadic prediction, ECML/PKDD 2014.

<sup>31</sup> Romero-Paredes and Torr, An embarrassingly simple approach to zero-shot learning, ICML 2015.

## Two-step zero-shot learning<sup>32 33</sup>



1,3	0,2	1,4	1,7	3,5	1,3	1,2
2	1,7	1,5	7,5	8,2	7,6	1,4
0,2	0	0,3	0,4	1,2	2,2	3,8
3,1	1,1	1,3	1,1	1,7	5,2	1,1
4,7	2,1	2,5	1,5	2,3	8,5	1,5

1,2	2,1	1,7	4,3	2,4	2,5	4,3
-----	-----	-----	-----	-----	-----	-----

<sup>32</sup> Pahikkala et al. A two-step approach for solving full and almost full cold-start problems in dyadic prediction, ECML/PKDD 2014.

<sup>33</sup> Romero-Paredes and Torr, An embarrassingly simple approach to zero-shot learning, ICML 2015.

## Two-step kernel ridge regression

- Kernel evaluations for new test instance:

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n))^T$$

$$\mathbf{g}(\mathbf{t}) = (g(\mathbf{t}, \mathbf{t}_1), \dots, g(\mathbf{t}, \mathbf{t}_q))^T$$

## Two-step kernel ridge regression

- Kernel evaluations for new test instance:

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n))^T$$

$$\mathbf{g}(\mathbf{t}) = (g(\mathbf{t}, \mathbf{t}_1), \dots, g(\mathbf{t}, \mathbf{t}_q))^T$$

- Step 1: prediction for  $\mathbf{x}$  on all the training targets

$$\mathbf{f}_T(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T A^{IT} = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y$$

## Two-step kernel ridge regression

- Kernel evaluations for new test instance:

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n))^T$$

$$\mathbf{g}(\mathbf{t}) = (g(\mathbf{t}, \mathbf{t}_1), \dots, g(\mathbf{t}, \mathbf{t}_q))^T$$

- Step 1: prediction for  $\mathbf{x}$  on all the training targets

$$\mathbf{f}_T(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T A^{IT} = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y$$

- Step 2: generalizing to new targets

$$f^{\text{TS}}(\mathbf{x}, \mathbf{t}) = \mathbf{g}(\mathbf{t})^T (\mathbf{G} + \lambda_t \mathbf{I})^{-1} \mathbf{f}_T(\mathbf{x})^T$$

## Two-step kernel ridge regression

- Kernel evaluations for new test instance:

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n))^T$$

$$\mathbf{g}(\mathbf{t}) = (g(\mathbf{t}, \mathbf{t}_1), \dots, g(\mathbf{t}, \mathbf{t}_q))^T$$

- Step 1: prediction for  $\mathbf{x}$  on all the training targets

$$\mathbf{f}_T(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T A^{IT} = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y$$

- Step 2: generalizing to new targets

$$\begin{aligned} f^{\text{TS}}(\mathbf{x}, \mathbf{t}) &= \mathbf{g}(\mathbf{t})^T (\mathbf{G} + \lambda_t \mathbf{I})^{-1} \mathbf{f}_T(\mathbf{x})^T \\ &= \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y (\mathbf{G} + \lambda_t \mathbf{I})^{-1} \mathbf{g}(\mathbf{t}) \end{aligned}$$

## Two-step kernel ridge regression

- Kernel evaluations for new test instance:

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_n))^T$$

$$\mathbf{g}(\mathbf{t}) = (g(\mathbf{t}, \mathbf{t}_1), \dots, g(\mathbf{t}, \mathbf{t}_q))^T$$

- Step 1: prediction for  $\mathbf{x}$  on all the training targets

$$\mathbf{f}_T(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T A^{IT} = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y$$

- Step 2: generalizing to new targets

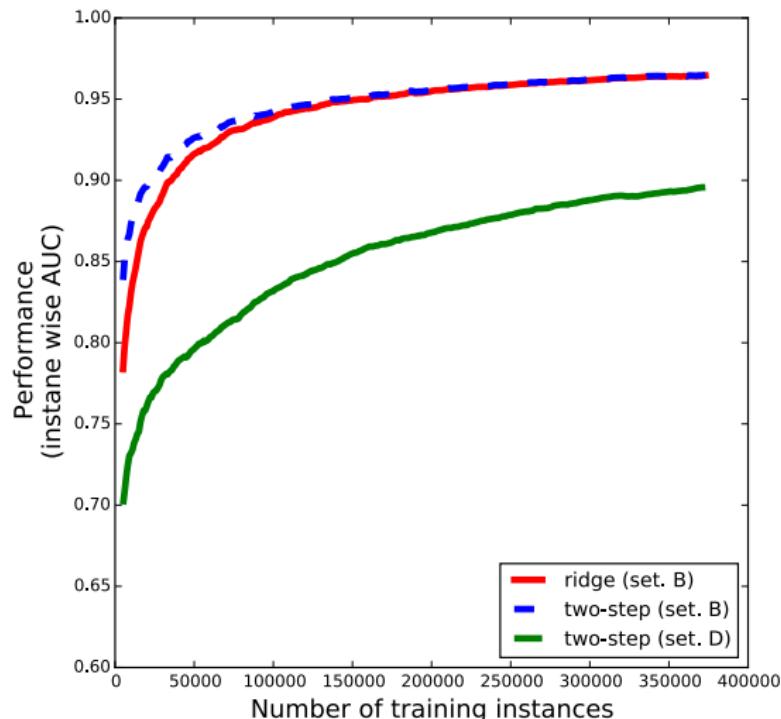
$$\begin{aligned} f^{\text{TS}}(\mathbf{x}, \mathbf{t}) &= \mathbf{g}(\mathbf{t})^T (\mathbf{G} + \lambda_t \mathbf{I})^{-1} \mathbf{f}_T(\mathbf{x})^T \\ &= \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda_d \mathbf{I})^{-1} Y (\mathbf{G} + \lambda_t \mathbf{I})^{-1} \mathbf{g}(\mathbf{t}) \\ &= \mathbf{k}(\mathbf{x})^T A^{\text{TS}} \mathbf{g}(\mathbf{t}) \\ &= \mathbf{w}^T (\phi(\mathbf{x}) \otimes \psi(\mathbf{t})) \end{aligned}$$

## In which situations does zero-shot learning work?

---

<sup>34</sup> M. Stock, Exact and efficient algorithms for pairwise learning, PhD thesis, 2017

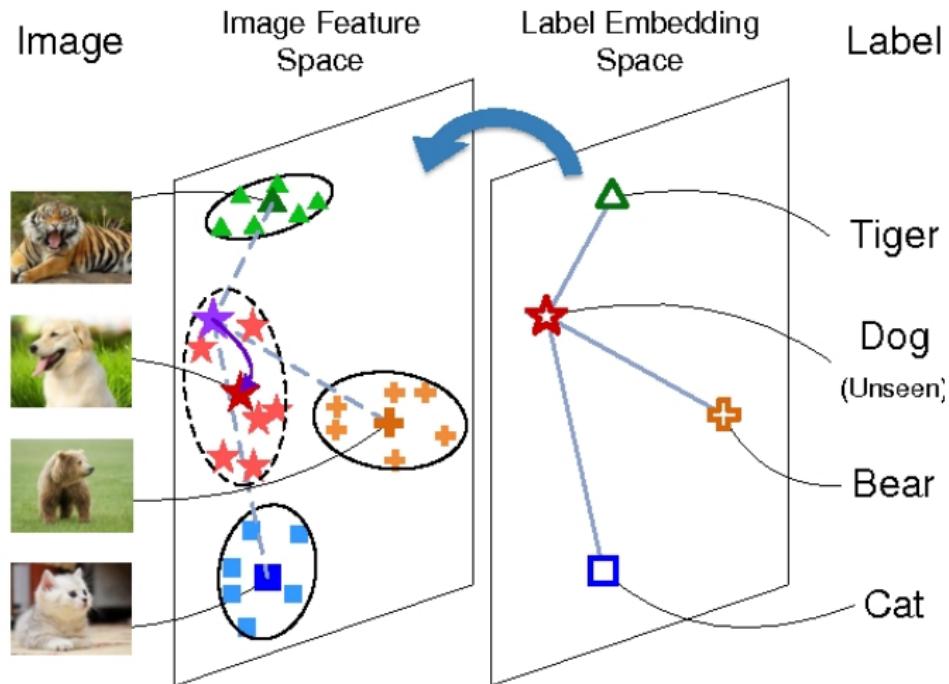
# In which situations does zero-shot learning work?



12,000 labels: from 5,000 to 350,000 instances<sup>34</sup>

<sup>34</sup> M. Stock, Exact and efficient algorithms for pairwise learning, PhD thesis, 2017

# Zero-shot learning in computer vision



# Zero-shot learning in computer vision

Pairwise model representation as before:

$$f(\mathbf{x}, \mathbf{t}) = \mathbf{w}^T (\phi(\mathbf{x}) \otimes \psi(\mathbf{t}))$$

Inference in a structured prediction fashion:

$$\hat{c}(\mathbf{x}) = \arg \max_{\mathbf{t} \in \mathcal{T}} f(\mathbf{x}, \mathbf{t})$$

Different optimization problems:

- Multi-class objective<sup>35</sup>
- Ranking objective<sup>36</sup>
- Regression objective<sup>37</sup>
- Canonical correlation analysis

Different model formulations:

- Linear embeddings
- Nonlinear embeddings

---

<sup>35</sup> Akata et al., Evaluation of Output Embeddings for Fine-Grained Image Classification, CVPR2015

<sup>36</sup> Frome et al., Devise: A deep visual-semantic embedding model, NIPS 2013

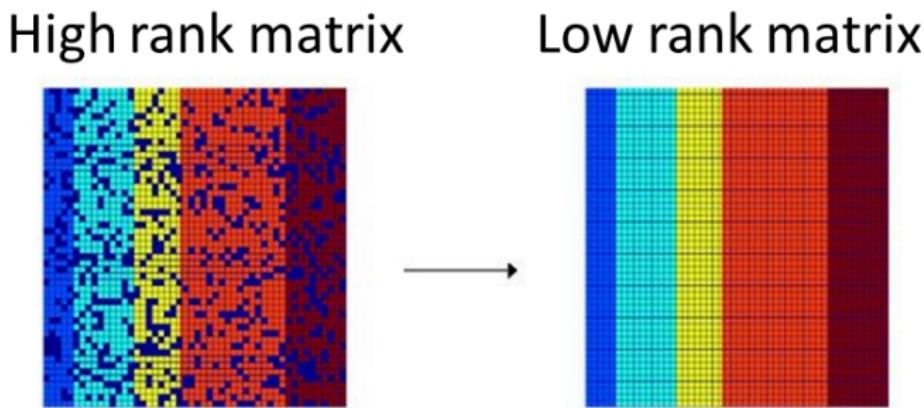
<sup>37</sup> Socher et al., g. Zero-shot learning through cross-modal transfer, NIPS 2013

# A unifying view on MTP methods



Group of methods	Applicable setting
Independent models	B and C
Similarity-enforcing methods	B and C
Relation-exploiting methods	B, C and D
Relation-constructing methods	B and C
Representation-exploiting methods	B, C and D
<b>Representation-constructing methods</b>	A, B and C

## Low-rank approximation in Settings B and C



Typically perform a low-rank approximation of the parameter matrix<sup>38</sup>:

$$\min_A \|Y - XA\|_F^2 + \lambda \text{rank}(A)$$

<sup>38</sup>Chen et al., A convex formulation for learning shared structures from multiple tasks, ICML 2009.

## Low-rank approximation in Settings B and C

- $A$ : parameter matrix of dimensionality  $p \times m$
- $p$ : the number of features
- $m$ : the number of targets
- Assume a low-rank structure of  $A$ :

$$U \quad \times \quad V \quad = \quad A$$

The diagram shows three matrices. Matrix  $U$  is a 3x3 grid of squares. Matrix  $V$  is a 3x3 grid of squares. Matrix  $A$  is a 3x3 grid of squares. Between  $U$  and  $V$  is a multiplication symbol ( $\times$ ). To the right of  $=$  is an approximation symbol ( $\approx$ ). This illustrates that  $U$  and  $V$  are multiplied to approximate matrix  $A$ .

- We can write  $A = VU$  and  $A\mathbf{x} = VU\mathbf{x}$
- $V$  is a  $p \times \hat{m}$  matrix
- $U$  is an  $\hat{m} \times m$  matrix
- $\hat{m}$  is the rank of  $A$

# Low-rank approximation in Settings B and C

## Overview of methods

- Popular for multi-output regression, multi-task learning and multi-label classification
- Linear as well as nonlinear methods
- Algorithms:
  - ▶ Principal component analysis<sup>39</sup>, Canonical correlation analysis<sup>40</sup>, Partial least squares
  - ▶ Singular value decomposition<sup>41</sup>, Alternating structure optimization<sup>42</sup>
  - ▶ Compressed sensing<sup>43</sup>, Output codes<sup>44</sup>, Landmark labels<sup>45</sup>, Bloom filters<sup>46</sup>, Auto-encoders<sup>47</sup>

---

<sup>39</sup> Weston et al., Kernel dependency estimation, NIPS 2002

<sup>40</sup> Multi-label prediction via sparse infinite CCA, NIPS 2009

<sup>41</sup> Tai and Lin, Multilabel classification with principal label space transformation, Neural Computation 2012

<sup>42</sup> Zhou et al., Clustered Multi-Task Learning Via Alternating Structure Optimization, NIPS 2011

<sup>43</sup> Hsu et al., Multi-label prediction via compressed sensing. NIPS 2009

<sup>44</sup> Zhang and Schneider, Multi-label Output Codes using Canonical Correlation Analysis, UAI 2011

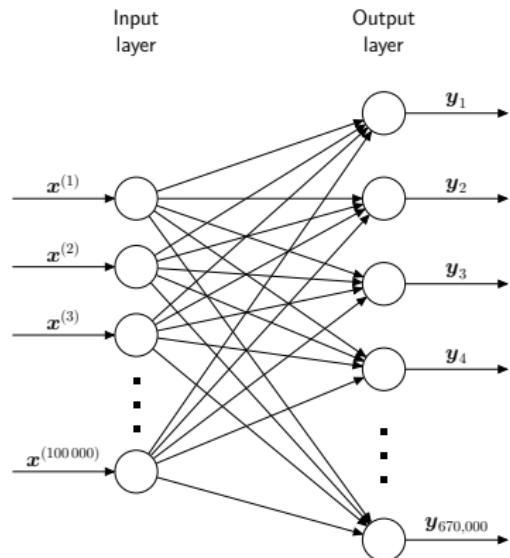
<sup>45</sup> Balasubramanian and Lebanon, The landmark selection method for multiple output prediction, ICML 2012

<sup>46</sup> Cissé et al., Robust bloom filters for large multilabel classification tasks, NIPS 2013

<sup>47</sup> Wicker et al., A nonlinear label compression and transformation method for multi-label classification using autoencoders, PAKDD 2016

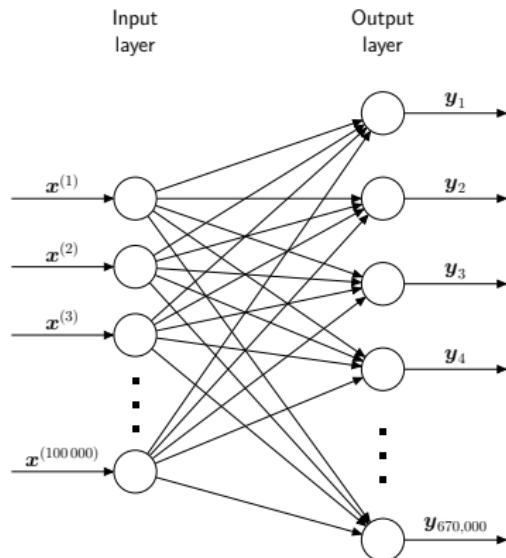
# Target embeddings in neural networks

- Shallow Networks - SVM
  - ▶ Direct mapping of input to output

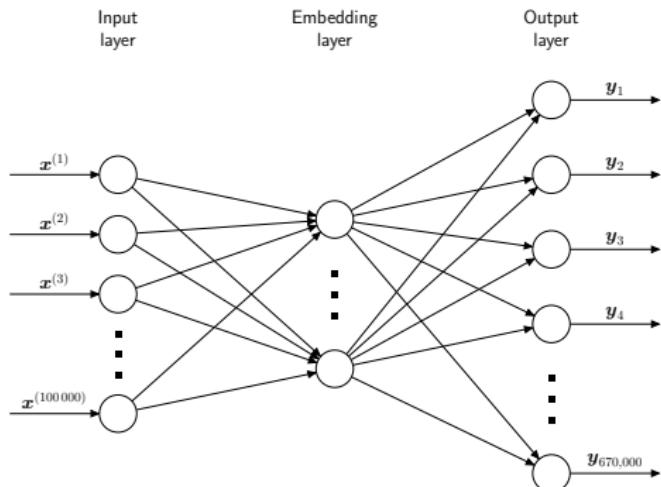


# Target embeddings in neural networks

- Shallow Networks - SVM
  - ▶ Direct mapping of input to output



- Label embedding
  - ▶ Mapping input to output via embedding layer



## Low-rank approximation in Setting A

Factorize the matrix  $Y$  instead of the parameter matrix  $A$ :

		Items									
		1	3	5		5	4				
		5		4				2	1	3	
		2.4									
Users		2	4	1	2	3	4	3	5		
		2	4	5		4			2		
		4	3	4	2				2	5	
		1	3	3		2			4		

~

		Items									
		.1	-.4	.2							
		-5	.6	.5							
Users		-2	.3	.5							
		1.1	2.1	.3							
		-.7	2.1	-2							
		-1	.7	.3							

•

Items											
.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-8	.7	.5	1.4	-3	-1	1.4	2.9	-.7	1.2	-1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Users

$$Y = U \times V$$

# Low-rank approximation in Setting A

## Overview of algorithms

- Nuclear norm minimization<sup>48</sup>
- Gaussian processes<sup>49</sup>
- Probabilistic methods<sup>50</sup>
- Spectral regularization<sup>51</sup>
- Non-negative matrix factorization<sup>52</sup>
- Alternating least-squares minimization<sup>53</sup>

---

<sup>48</sup> Candes and Recht, Exact low-rank matrix completion via convex optimization. Foundations of Computational Mathematics 2008

<sup>49</sup> Lawrence and Urtasun, Non-linear matrix factorization with Gaussian processes, ICML 2009

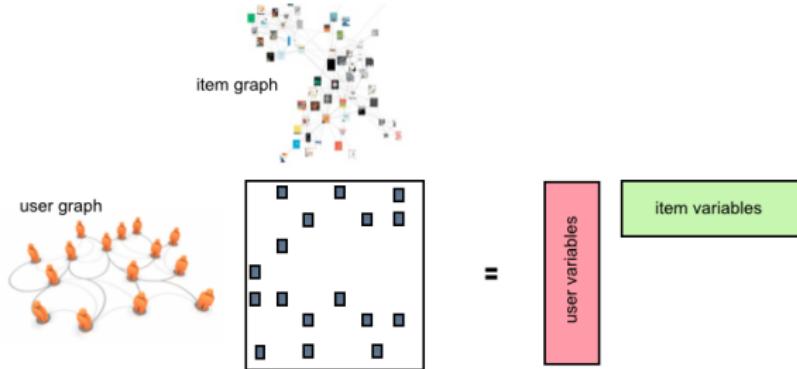
<sup>50</sup> Shan and Banerjee, Generalized probabilistic matrix factorizations for collaborative filtering, ICDM 2010

<sup>51</sup> Mazumder et al., Spectral regularization algorithms for learning large incomplete matrices., JMLR 2010

<sup>52</sup> Gaujoux and Seoighe, A flexible R package for nonnegative matrix factorization. BMC bioinformatics 2010

<sup>53</sup> Jain et al., Low-rank matrix completion using alternating minimization, ACM Symposium on Theory of Computing 2013

# Matrix factorization with side information for Setting A



- Construct **implicit** features  $(\mathbf{x}^I, \mathbf{t}^I)$  for users and items with matrix factorization methods
- Exploit **explicit** features  $(\mathbf{x}^E, \mathbf{t}^E)$  (a.k.a. side information)
- Concatenate:

$$\mathbf{x}^C = (\mathbf{x}^I, \mathbf{x}^E), \quad \mathbf{t}^C = (\mathbf{t}^I, \mathbf{t}^E)$$

- Apply methods that we have seen before<sup>5455</sup>:

$$f(\mathbf{x}^C, \mathbf{t}^C) = \mathbf{w}^T (\phi(\mathbf{x}^C) \otimes \psi(\mathbf{t}^C))$$

<sup>54</sup> Menon and Elkan, A log-linear model with latent features for dyadic prediction, ICDM 2010

<sup>55</sup> Volkovs and Zemel, Collaborative filtering with 17 parameters, NIPS 2012

## Hybrid matrix factorization for Setting A

Basilico and Hofmann, 2004; Abernethy et al, 2008; Adams et al, 2010;  
Fang and Si, 2011; Zhou et al, 2011a; Menon and Elkan, 2011; Zhou et al,  
2012b).

# When is it useful to construct target representations?

Does not work well in extreme multi-label classification<sup>56</sup>:

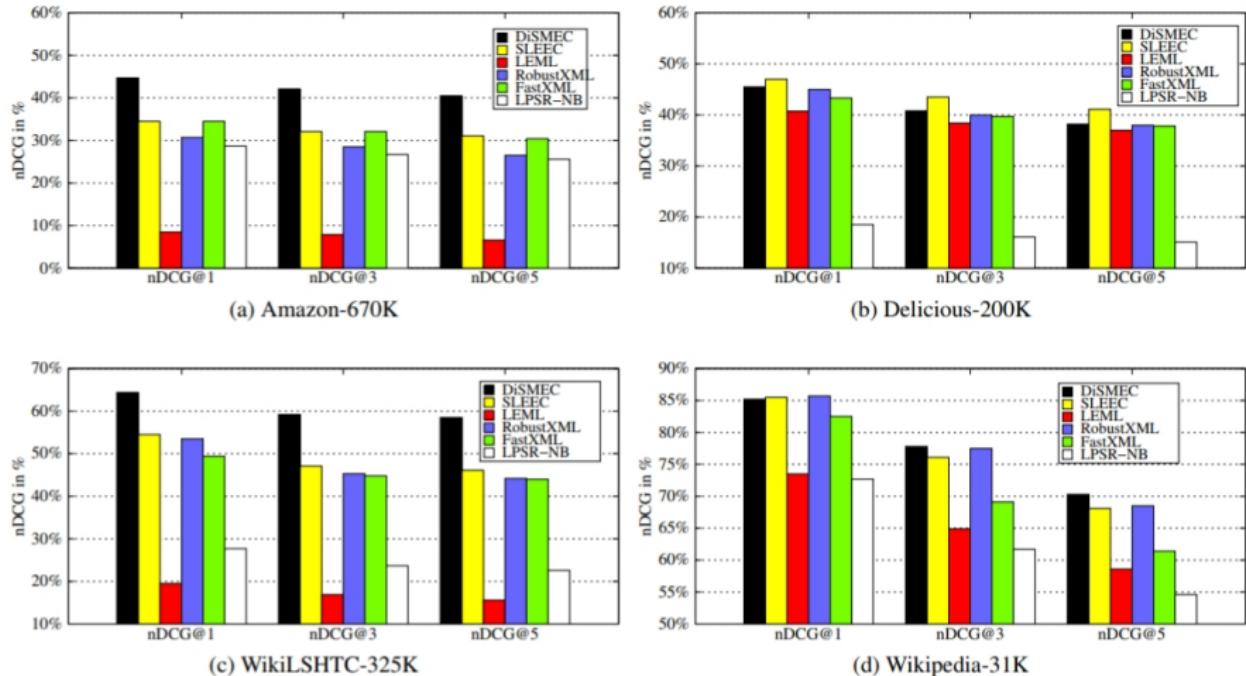


Figure 3: nDCG@k for k=1, 3 and 5

<sup>56</sup> Babbar and Schölkopf, DISMEC: Distributed Sparse Machines for Extreme Multi-label classification, WSDM 2017

# When is it useful to construct target representations?

## SVD interpretation

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

Diagram illustrating the Singular Value Decomposition (SVD) of matrix  $A$ :

- Representation of instance (or feature)**: Matrix  $\mathbf{U}$  is shown as a gray rectangle. A vertical line labeled  $j$  indicates a specific column, representing the representation of an instance or feature.
- Representation of target**: Matrix  $\mathbf{V}^T$  is shown as a gray rectangle. A vertical line labeled  $i$  indicates a specific row, representing the representation of a target.
- Matrix  $A$** : Matrix  $\mathbf{A}$  is shown as a gray rectangle.
- Sigma matrix  $\Sigma$** : Matrix  $\Sigma$  is shown as a gray rectangle with diagonal lines representing non-zero singular values  $\sigma_1, \sigma_2, \dots, \sigma_I$ .

Dimensions below the matrices:

- $\mathbf{A}$ :  $p \times m$
- $\mathbf{U}$ :  $p \times p$
- $\Sigma$ :  $p \times m$
- $\mathbf{V}^T$ :  $m \times m$

- $\sigma_1, \sigma_2, \dots$ : singular values of  $A$
- Rank of  $A$  = number of non-zero **singular values**
- **High rank** when a lot of singular values differ from zero
- **Low rank** when a lot of singular values are zero
- Singular values **give insight** in what can be gained

# Overview of this tutorial

- 1 A unifying view on MTP problems
- 2 A unifying view on MTP methods
- 3 Loss functions in multi-target prediction
- 4 Conclusions

# Conclusions

- Multi-target prediction is an active field of research that connects different types of machine learning problems
- In the corresponding subfields of machine learning, problems have typically been solved in isolation, without establishing connections between methods
- Two-step zero-shot learning is a simple MTP method with a lot of interesting properties

**Upcoming paper:**

**Waegeman et al.**

**Multi-Target Prediction:**

**A Unifying View on Problems and Methods**