Unifying Perspectives on Knowledge Sharing:
From Atomic to Parameterised Domains and Tasks

Task-CV @ ECCV 2016

Timothy Hospedales
University of Edinburgh & Queen Mary University of London

With

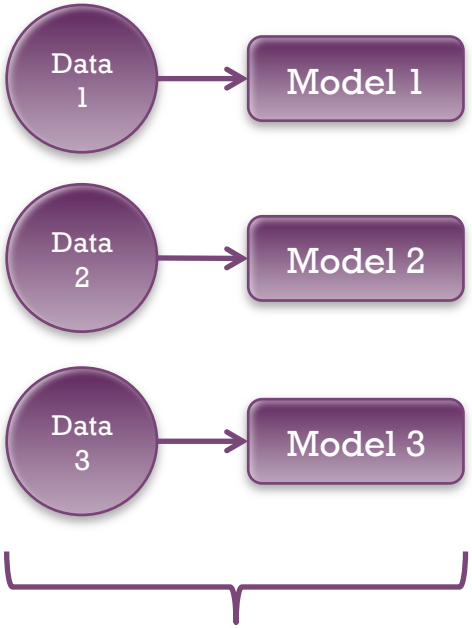
Yongxin Yang
Queen Mary University of London

Today's Topics

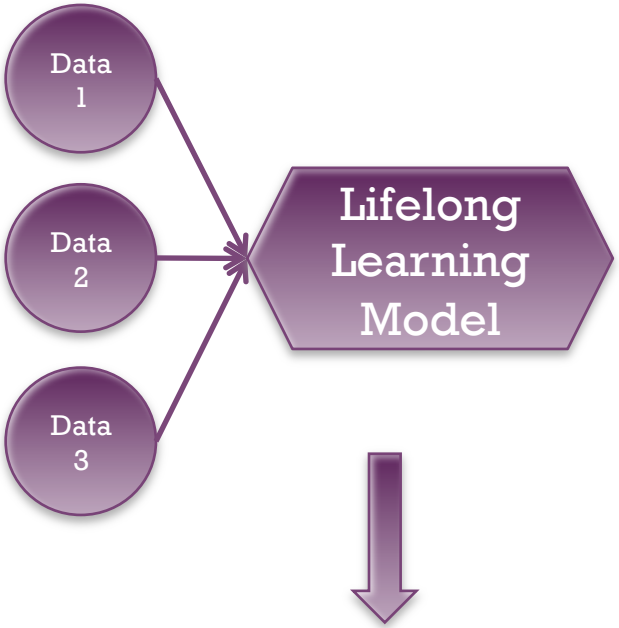
- Distributed definitions of task/domains, and different problem settings that arise.

- A flexible approach to task/domain transfer
  - Generalizes existing approaches
  - Generalizes multiple problem settings
  - Covers shallow and deep models

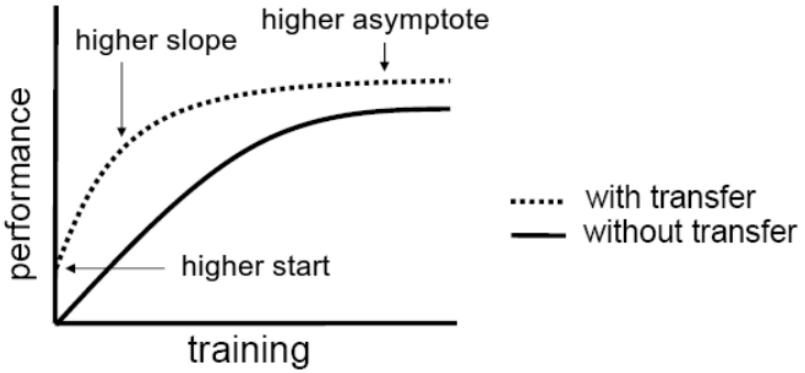
Why Transfer Learning?



IID
Tasks or Domains



But…. Humans seem to generalize across tasks
E.g., Crawl => Walk => Run => Scooter => Bike => Motorbike => Driving.



Taxonomy of Research Issues

Sharing Setting

- Sequential / One-way
- Multi-task
- Life-long learning

Sharing Approach

- Model-based
- Instance-based
- Feature-based

Balancing Challenge

- Positive Transfer Strength
- Negative Transfer Robustness

Labeling assumption

- Supervised
- Unsupervised

Transfer Across:

- Task Transfer
- Domain Transfer

Feature/Label Space

- Homogeneous
- Heterogeneous

Taxonomy of Research Issues

Sharing Setting

- Sequential / One-way
- Multi-task
- Life-long learning

Sharing Approach

- Model-based
- Instance-based
- Feature-based

Balancing Challenge

- Positive Transfer Strength
- Negative Transfer Robustness

Labeling assumption

- Supervised
- Unsupervised

Transfer Across:

- Task Transfer
- Domain Transfer

Feature/Label Space

- Homogeneous
- Heterogeneous

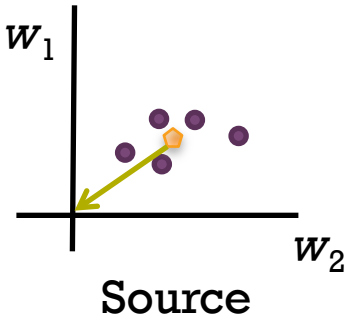
Overview

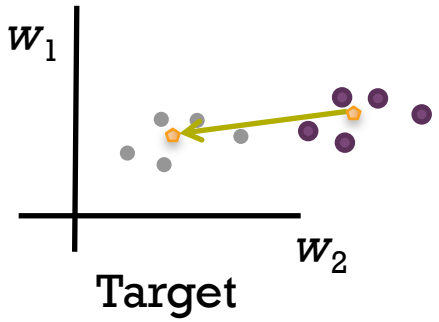
- A review of some classic methods
- A general framework
- Example problems and settings
- Going deeper
- Open questions

Some Classic Methods – 1
Model Adaptation

An example of simple sequential transfer:

- Learn a source task: $y = f_s(\mathbf{x}, \mathbf{w}_s)$ $\quad \min_{\mathbf{w}_s} \sum_i \left| y_i - \mathbf{w}_s^T \mathbf{x}_i \right| + \lambda \mathbf{w}_s^T \mathbf{w}_s$
- Learn a target new task: $y = f_t(\mathbf{x}, \mathbf{w})$ $\quad \min_{\mathbf{w}} \sum_i \left| y_i - \mathbf{w}^T \mathbf{x}_i \right| + \lambda (\mathbf{w} - \mathbf{w}_s)^T (\mathbf{w} - \mathbf{w}_s)$
- Regularize new task toward old task
  - (…rather than toward zero)





E.g., Yang, ACM MM, 2007

Some Classic Methods – 1
Model Adaptation

An example of simple sequential transfer:

- Learn a target new task: $y = f_t(\mathbf{x}, \mathbf{w})$ $\min_{\mathbf{w}} \sum_i \left| y_i - \mathbf{w}^T \mathbf{x}_i \right| + \lambda (\mathbf{w} - \mathbf{w}_s)^T (\mathbf{w} - \mathbf{w}_s)$

- Limitations:
  - ✘ Assumes relatedness of source task
  - ✘ Only sequential, one-way transfer

E.g., Yang, ACM MM, 2007

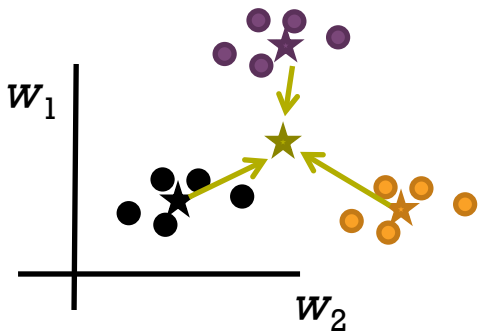
Some Classic Methods – 2
Regularized Multi-Task

An example of simple multi-task transfer:

- Learn a set of tasks: $\{\mathbf{x}_{i,t}, y_{i,t}\} \Rightarrow \{y = f_t(\mathbf{x}, \mathbf{w}_t)\}$

$$\min_{\substack{\mathbf{w}_0, \mathbf{w}_t \\ t=1..T}} \sum_{i,t} \left| y_{i,t} - \mathbf{w}_t^T \mathbf{x}_{i,t} \right| + \lambda (\mathbf{w}_t - \mathbf{w}_0)^T (\mathbf{w}_t - \mathbf{w}_0)$$

- Regularize each task towards mean of all tasks:



E.g., Evgeniou & Pontil, KDD'04
E.g., Salakhutdinov, CVPR'11
E.g., Khosla, ECCV'12

Some Classic Methods – 2
Regularized Multi-Task

An example of simple multi-task transfer:

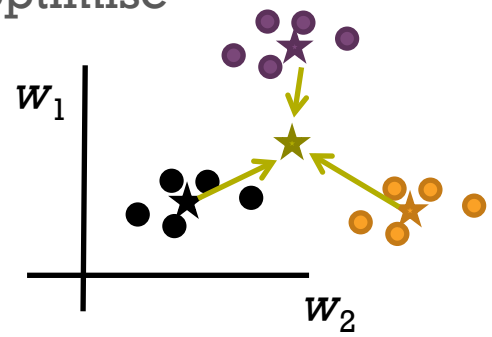
- Learn a set of tasks: $\{\mathbf{x}_{i,t}, y_{i,t}\} \Rightarrow \{y = f_t(\mathbf{x}, \mathbf{w}_t)\}$

$$\min_{\substack{\mathbf{w}_0, \mathbf{w}_t \\ t=1..T}} \sum_{i,t} \left| y_{i,t} - \mathbf{w}_t^T \mathbf{x}_{i,t} \right| + \lambda (\mathbf{w}_t - \mathbf{w}_0)^T (\mathbf{w}_t - \mathbf{w}_0)$$

Or….

$$\min_{\substack{\mathbf{w}_0, \mathbf{w}_t \\ t=1..T}} \sum_{i,t} \left| y_{i,t} - (\mathbf{w}_t + \mathbf{w}_0)^T \mathbf{x}_{i,t} \right|$$

- Summary:
  - ✔ Now multi-task
  - ✗ Tasks and their mean are inter-dependent: jointly optimise
  - ✗ Still assumes all tasks are (equally) related

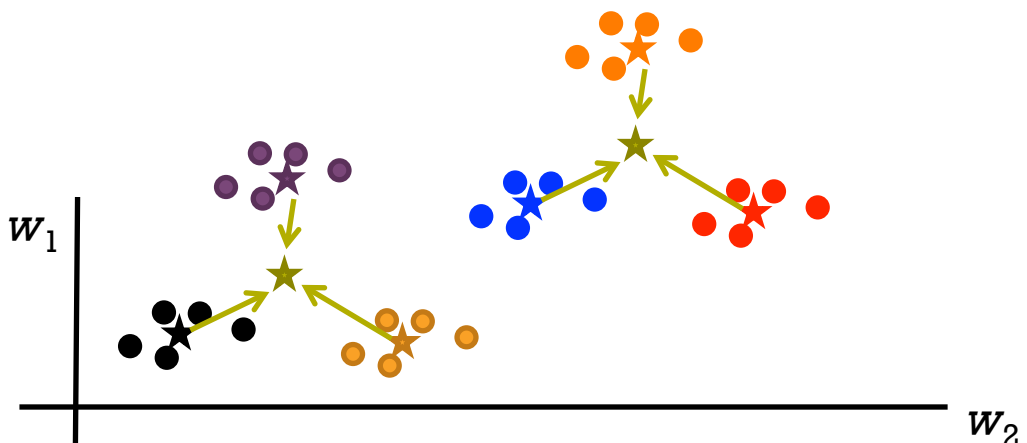


Some Classic Methods – 3
Task Clustering

Relaxing relatedness assumption through task clustering

- Learn a set of tasks: $\{\mathbf{x}_{i,t}, y_{i,t}\} \Rightarrow \{y = f_t(\mathbf{x}, \mathbf{w}_t)\}$
- Assume tasks form K similar groups:
  - Regularize task towards nearest group

$$\min_{\substack{\mathbf{w}_k, \mathbf{w}_t \\ k=1..K, t=1..T}} \sum_{i,t} \left| y_{i,t} - \mathbf{w}_t^T \mathbf{x}_{i,t} \right| + \min_{k'} \lambda (\mathbf{w}_t - \mathbf{w}_{k'})^T (\mathbf{w}_t - \mathbf{w}_{k'})$$



E.g., Evgeniou et al, JMLR, 2005
E.g., Kang et al, ICML, 2011

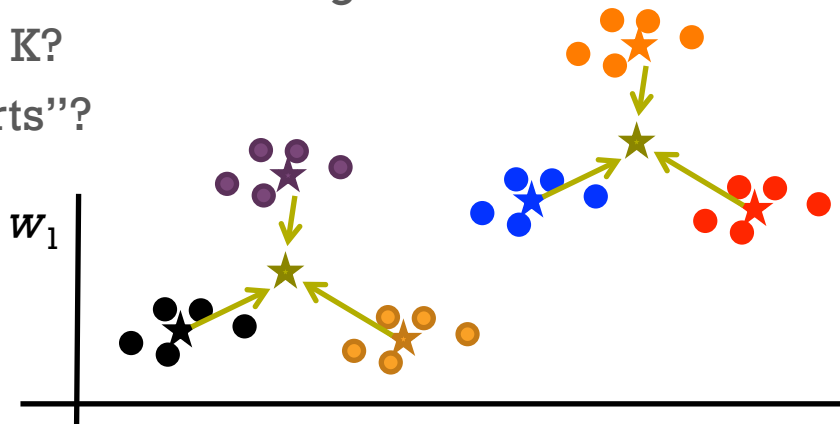
Some Classic Methods – 3
Task Clustering

Multi-task transfer without assuming relatedness

- Assume tasks form similar groups:

$$\min_{\substack{\mathbf{w}_k,\mathbf{w}_t \\ k=1..K,t=1..T}} \sum_{i,t} \left| y_{i,t} - \mathbf{w}_t^T \mathbf{x}_{i,t} \right| + \min_{k'} \lambda (\mathbf{w}_t - \mathbf{w}_{k'})^T (\mathbf{w}_t - \mathbf{w}_{k'})$$

- Summary:
  - ✓ Doesn't require all tasks related => More robust to negative transfer
  - ✓ Benefits from "more specific" transfer
  - ✗ What about task specific/task independent knowledge?
  - ✗ How to determine number of clusters K?
  - ✗ What if tasks share at the level of "parts"?
  - ✗ Optimization is hard



Some Classic Methods – 4
Task Factoring

- Learn a set of tasks $\{\mathbf{x}_{i,t}, y_{i,t}\} \Rightarrow \{y = f_t(\mathbf{x}, \mathbf{w}_t)\}$
- Assume related by a factor analysis / latent task structure.
- Notation: Input now triples: $\{\mathbf{x}_i, y_i, \mathbf{z}_i\}$ ← Binary task indicator vector
- STL, weight stacking notation: $y = f_t(x, W) = W_{(t,:)}^T \mathbf{x} = \left(W\mathbf{z}\right)^T \mathbf{x}$

  $$\Rightarrow \min_W \sum_i \left| y_i - \left(W\mathbf{z}_i\right)^T \mathbf{x}_i \right| + \lambda \left|W\right|_2^2$$

- Factor Analysis-MTL:

$$y = \left(W\mathbf{z}\right)^T \mathbf{x} = \left(PQ\mathbf{z}\right)^T \mathbf{x}$$

$$\Rightarrow \min_{P,Q} \sum_i \left| y_i - \left(PQ\mathbf{z}_i\right)^T \mathbf{x}_i \right| + \lambda \left|P\right| + \omega \left|Q\right|$$

E.g., Kumar, ICML'12
E.g., Passos, ICML'12

Some Classic Methods – 4
Task Factoring

- Learn a set of tasks $\{\mathbf{x}_i, y_i, \mathbf{z}_i\}$ ⇒ $y = f_t(\mathbf{x}, W)$
  - Assume related by a factor analysis / latent task structure.
- Factor Analysis-MTL: $y = \mathbf{w}_t^T \mathbf{x} = (W\mathbf{z})^T \mathbf{x} = (PQ\mathbf{z})^T \mathbf{x}$
- What does it mean? $\min_{P,Q} \sum_i \left| y_i - (PQ\mathbf{z}_i)^T \mathbf{x}_i \right| + \lambda |P| + \omega |Q|$
  - W: DxK matrix of all task parameters
  - P: DxK matrix of basis/latent tasks
  - Q: KxT matrix of low-dimensional task models
  - => Each task is a low-dimensional linear combination of basis tasks.

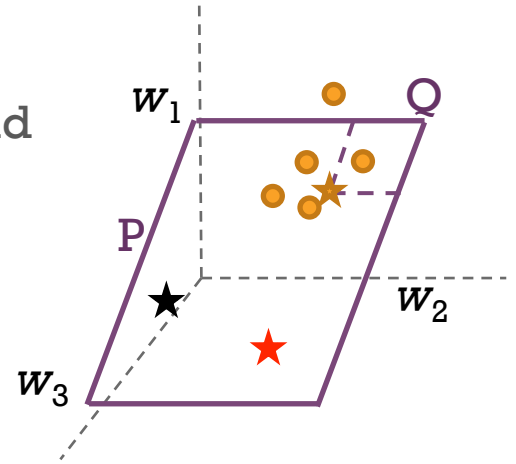
Some Classic Methods – 4
Task Factoring

- Learn a set of tasks $\{\mathbf{x}_i, y_i, \mathbf{z}_i\}$ ⇒ $y = f_t(\mathbf{x}, W)$
  - Assume related by a factor analysis / latent task structure.
- What does it mean?

$$y = \mathbf{w}_t^T \mathbf{x} = \left(W\mathbf{z}\right)^T \mathbf{x} = \left(PQ\mathbf{z}\right)^T \mathbf{x}$$

  - z: (1-hot binary) Activates a column of Q
  - P: DxK matrix of basis/latent tasks
  - Q: KxT matrix of task models
  - => Tasks lie on a low-dimensional manifold
  - => Knowledge sharing by jointly learning manifold

$$\min_{P,Q} \sum_i \left| y_i - \left(PQ\mathbf{z}_i\right)^T \mathbf{x}_i \right| + \lambda |P| + \omega |Q|$$



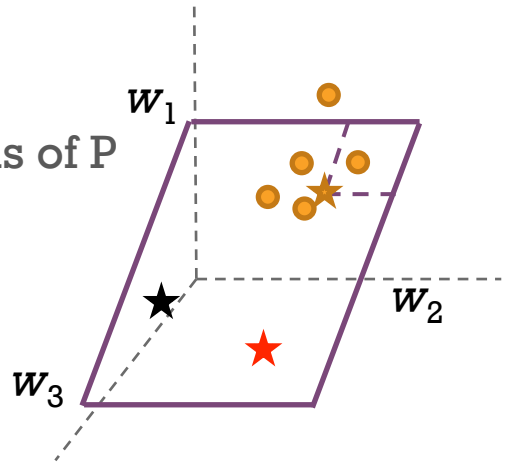
  - P: Specify the manifold
  - Q: Each task's position on the manifold

Some Classic Methods – 4
Task Factoring

- Summary:
  - Tasks lie on a low-dimensional manifold
  - Each task is a low-dimensional linear combination of basis tasks.
  - ✓ Can flexibly share or not share:
    - Two Q cols (tasks) similarity.
  - ✓ Can share piecewise:
    - Two Q cols (tasks) similar in some rows only
  - ✓ Can represent globally shared knowledge:
    - Uniform row in Q => all tasks activate same basis of P

$$y = \mathbf{w}_t^T \mathbf{x} = \left(W\mathbf{z}\right)^T \mathbf{x} = \left(PQ\mathbf{z}\right)^T \mathbf{x}$$

$$\min_{P,Q} \sum_i \left| y_i - \left(PQ\mathbf{z}_i\right)^T \mathbf{x}_i \right| + \lambda |P| + \omega |Q|$$

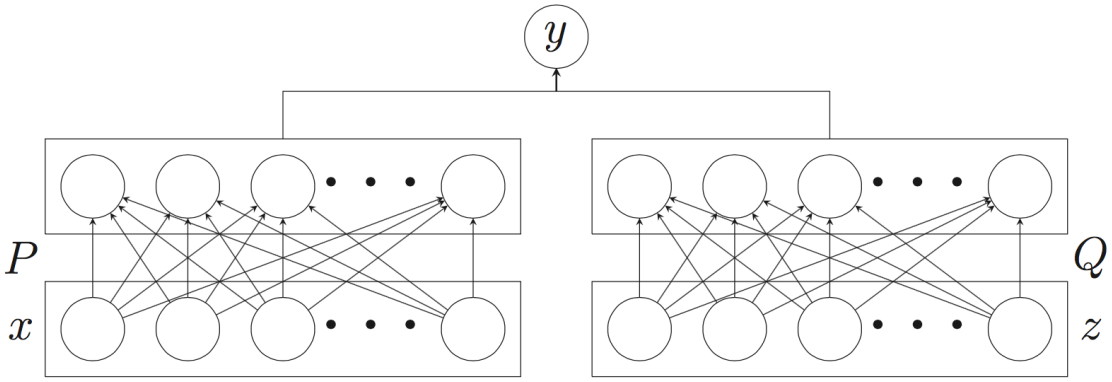


Overview

- A review of some classic methods
- A general framework
- Example problems and settings
- Going deeper
- Open questions

MTL Transfer as a Neural Network

- Consider a two sided neural network:
  - Left: Data input x.
  - Right: Task indicator z.
  - Output unit y: Inner product of representations
- Equivalent to: Task Regularization [Evgeniou KDD'04], if:
  - Q = W: (trainable) FC layer. P: (fixed) identity matrix.
  - z: 1-hot task encoding plus a bias bit => The shared knowledge
  - Linear activation



$$\min_{\substack{\mathbf{w}_0,\mathbf{w}_t \\ t=1..T}} \sum_{i,t} \left| y_{i,t} - \left(\mathbf{w}_t + \mathbf{w}_0\right)^T \mathbf{x}_{i,t} \right|$$

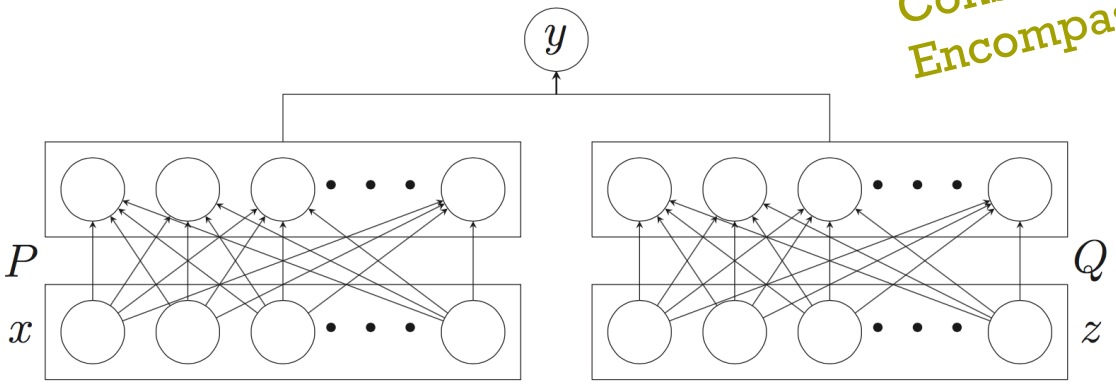
$$y = \left(\mathbf{w}_t + \mathbf{w}_0\right)^T \mathbf{x}$$

[Yang & Hospedales, ICLR'15]

MTL Transfer as a Neural Network

- Consider a two sided neural network:
  - Left: Data input x.
  - Right: Task indicator z.
  - Output unit y: Inner product of representation on each side.
- Equivalent to: Task Factor Analysis [Kumar, ICML'12, GO-MTL] if:
  - Train FC layers P&Q
  - z: 1-hot task encoding
  - Linear activation

Constraining task description/parameters:
Encompass: 5+ classic MTL/MDL approaches!



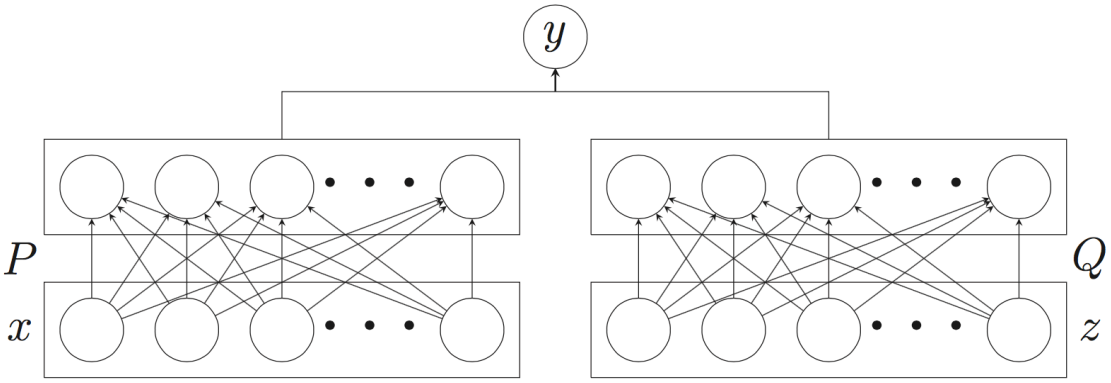
$$y = \left(W\mathbf{z}\right)^T \mathbf{x}$$

$$\min_{P,Q} \sum_i \left| y_i - \left(PQ\mathbf{z}_i\right)^T \mathbf{x}_i \right|$$

$$= \min_{P,Q} \sum_i \left| y_i - \left(P\mathbf{x}_i\right)\left(Q\mathbf{z}_i\right)^T \right|$$

MTL Transfer as a Neural Network: Interesting things

- Interesting things:
  - Generalizes many existing frameworks…
  - Can do regression & classification (activation on y).
  - Can do multi-task and multi-domain.
  - As neural network, left side X can be any CNN and train end-to-end



x: Data

z: Task/Domain-ID

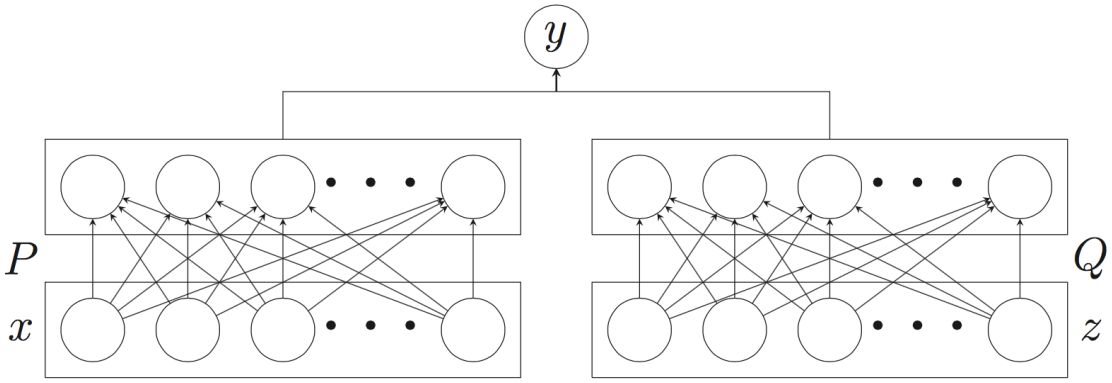
$$y = \left(W\mathbf{z}\right)^T \mathbf{x}$$

$$\min_{P,Q} \sum_i \left| y_i - \left(P\mathbf{x}_i\right)\left(Q\mathbf{z}_i\right)^T \right|$$

MTL Transfer as a Neural Network: Interesting things

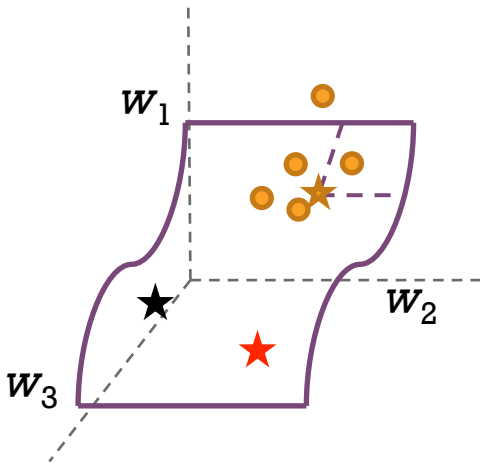
Interesting things:

- Non-linear activation on hidden layers:
  - Have representation learning on both task and data.
  - Exploit a non-linear task subspace.
    - CF GO-MTL's linear task subspace.
  - Final classifier can be non-linear in feature space.



x: Data

z: Task/Domain-ID



$$y = \sigma(P\mathbf{x})\sigma(Q\mathbf{z})^T$$

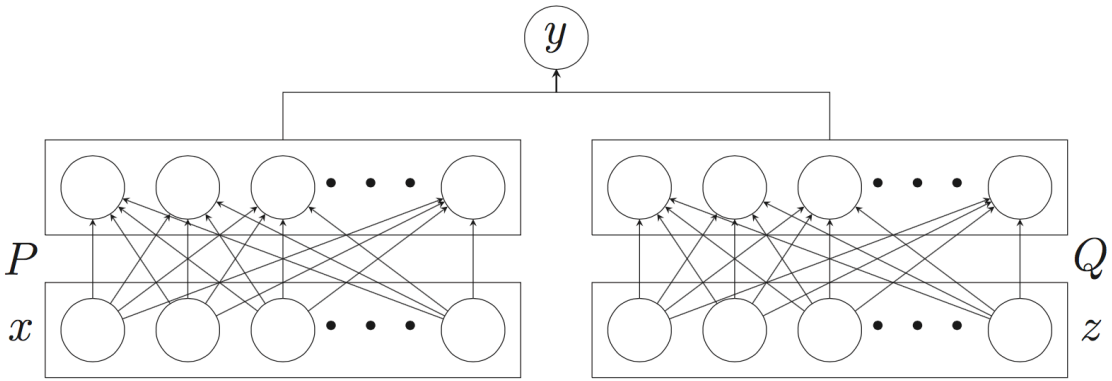
$$\min_{P,Q} \sum_i \left| y_i - \sigma(P\mathbf{x}_i)\sigma(Q\mathbf{z}_i)^T \right|$$

Overview

- A review of some classic methods
- A general framework
- Example problems and settings
- Going deeper
- Open questions

From Indexes to Task and Domain Descriptors

- Classic Task & Domain transfer:
  - Index atomic tasks/domains. (z is 1-of-T encoding)
- In many cases have task/domain metadata.
  - Let z be a more general task descriptor.
  - Distributed representation z : provides a "prior" for how to share latent tasks
  - E.g., Object recognition: Task = object category
    - Improve MTL learning with descriptor: z = attribute bit-string, wordvector



$$y = \sigma(P\mathbf{x})\sigma(Q\mathbf{z})^T$$

$$\min_{P,Q} \sum_i \left| y_i - \sigma(P\mathbf{x}_i)\sigma(Q\mathbf{z}_i)^T \right|$$

MTL With Informative Tasks

MTL: Informative Tasks

"Tiger"

Task: Furry, Carnivore
black, brown: [1,1,0,1,0,0]

"Panda"

Task: Furry, Vegetarian
black, white: [1,0,1,1,0,1]

Sharing "Prior"

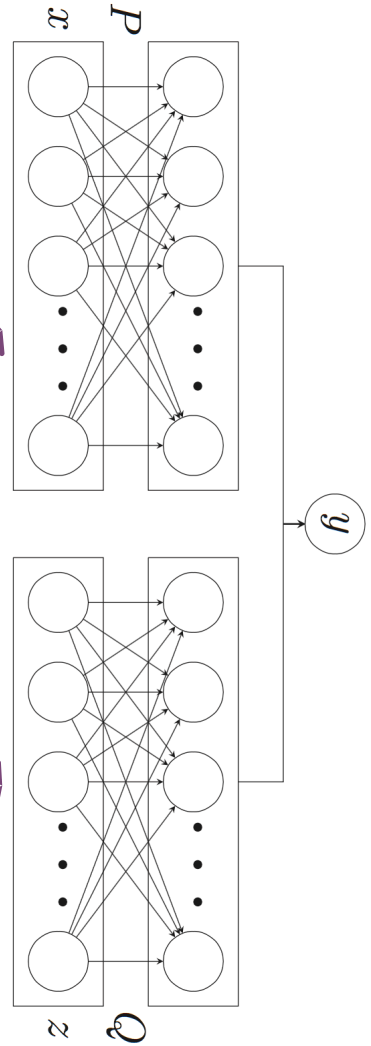
MTL: Atomic Tasks

"Tiger"

Task: ID=[0,1]

"Panda"

Task: ID=[1,0]

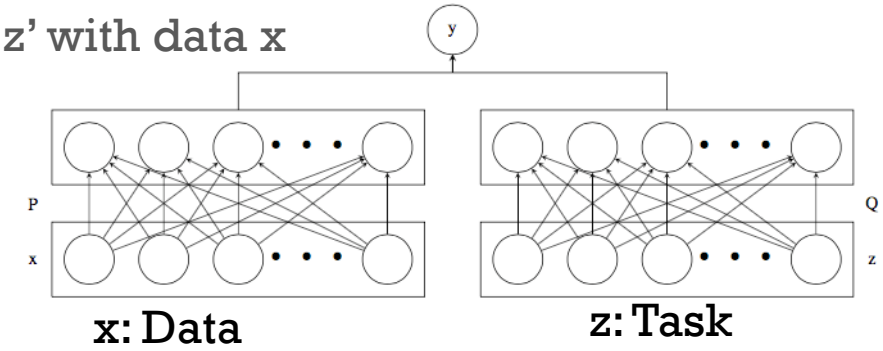


[Yang & Hospedales, ICLR'15]

Neural Net Zero-Shot Learning

Task-Description MTL gets ZSL for free

- Conventional MTL:
  - y = f(x,z): 1/0 for 1-v-all. x: data. z: category index
- MTL with task description
  - y = f(x,z): 1/0 for 1-v-all. x: data. z: category description (e.g., attributes).
- From descriptor driven MTL to ZSL:
  - With this framework you don't have to have seen a task to recognise it.
- ZSL Pipeline:
  - Train: 1-v-all to accept matched data & descriptors, reject mismatched.
  - Test: Compare novel task descriptors z' with data x
    - Pick $z^* = \text{argmax}_{z'} f(x,z')$



Task-Description MTL gets ZSL for free (Just describe new task)

Train

"Tiger"

Task: Furry, Carnivore
black, brown: [1,1,0,1,0,0]

"Panda"

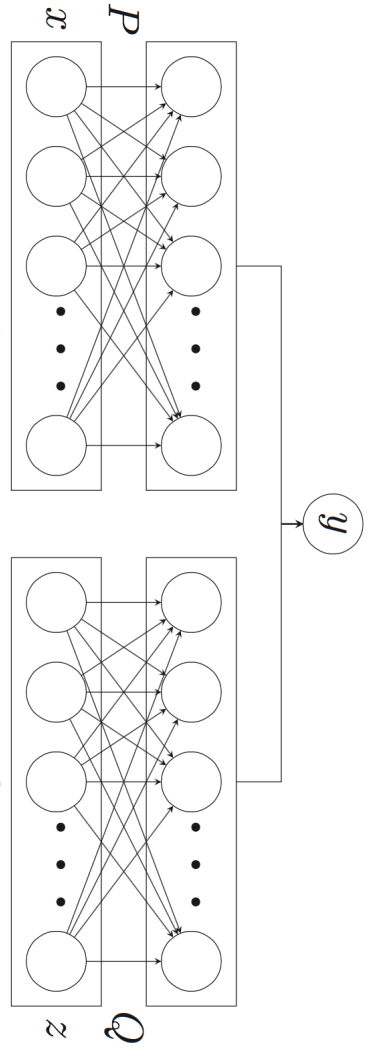
Task: Furry, Vegetarian
black, white: [1,0,1,1,0,1]

Test

"Black Lepoard"

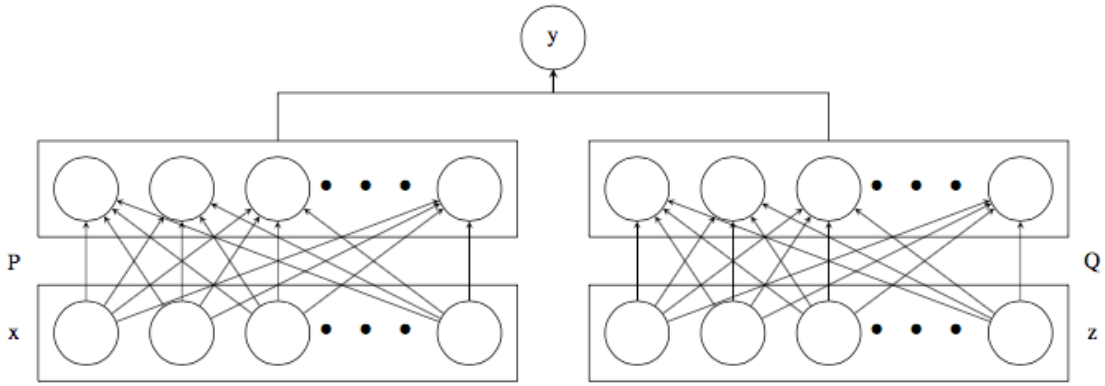
Task: Furry, Carnivore,
Black: [1,1,0,1,0,0]

Or wordvec, etc



From Indexes to Task and Domain Descriptors

- Classic Task & Domain transfer:
  - Index tasks/domains. (z is 1-hot encoding)
- In many cases have task/domain metadata.
  - Let z be a more general domain descriptor.
  - Distributed representation z : provides a prior for sharing the latent domains
  - Gait-based identification: z = camera view angle, distance, floor type
  - Audio-recognition: z=microphone type, room type, background noise type.



x: Data

z: Domain

$$y = \sigma\left(P\mathbf{x}\right)\sigma\left(Q\mathbf{z}\right)^T$$

$$\min_{P,Q} \sum_i \left| y_i - \sigma\left(P\mathbf{x}_i\right)\sigma\left(Q\mathbf{z}_i\right)^T \right|$$

Multi-Domain Learning with Descriptors: Example

Surveillance dataset
Hoffman CVPR'14

Domain = 1 | Domain = 2 | Domain = 3 | Conventional DA/MDL

Time = 1 | Time= 3 | Time = 17

Temporal Domain Evolution
(Lampert CVPR'15,
Hoffman CVPR'14)

Degree of Similarity vs Type of Similarity

Evening, Summer
6PM, Weekday

Night, Summer
1AM, Weekend

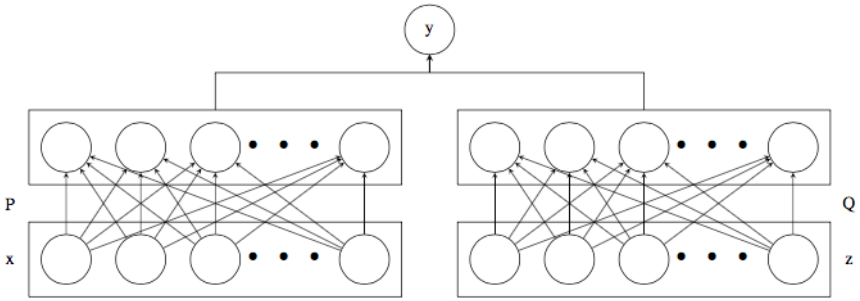
Day, Winter
6PM, Weekend

Richer Domain
Descriptions

[Yang & Hospedales, ICLR'15, CVPR'16, arXiv '16]

Zero-Shot Domain Adaptation:
A New Problem!

- Zero-Shot Domain Adaptation:
  - Analogy of ZSL but solve a novel domain rather than task.
- Pipeline:
  - Train: A few domains with descriptors.
  - Test:
    - "Calibrate" a new domain by input descriptor
    - => Immediate high accuracy recognition.



Zero-Shot Domain Adaptation: Car Type Recognition

Domain Factor 2: Decade

Domain Factor 1: View Angle

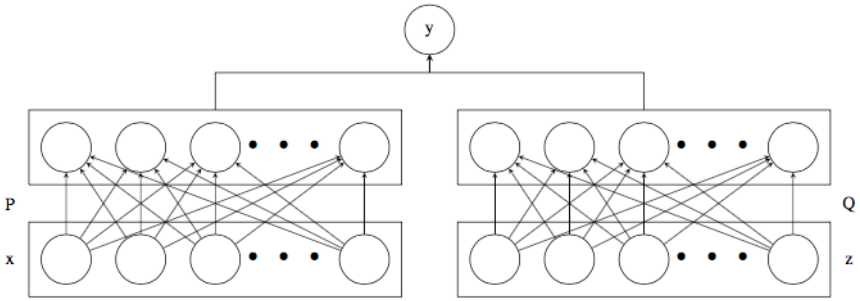
Test

Train

[Yang & Hospedales, ICLR'15, CVPR'16, arXiv '16]

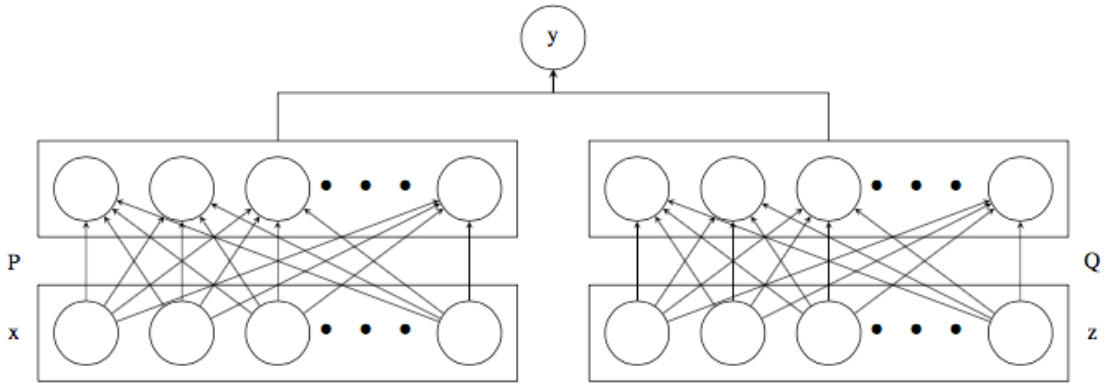
Zero-Shot Domain Adaptation:
A New Problem!

- Zero-Shot Domain Adaptation:
  - Analogy of ZSL but solve a novel domain rather than task.
- ZSDA Contrast: Domain Adaptation
  - ZSDA has no target domain data, either Labeled/Unlabeled
  - ZSDA has a target domain description
- ZSDA Contrast: Domain Generalization
  - ZSDA Should outperform DG
  - …due to leverage target domain description



From Indexes to Task and Domain Descriptors

- Interesting Things:
  - Can we unify Task/Domain sharing for synergistic MTL+MDL?
  - E.g., Digit Recognition
    - Task: Digits 0…9.
    - Domain: MNIST/USPS/SVHN.
- => Simultaneous MTL + MDL?



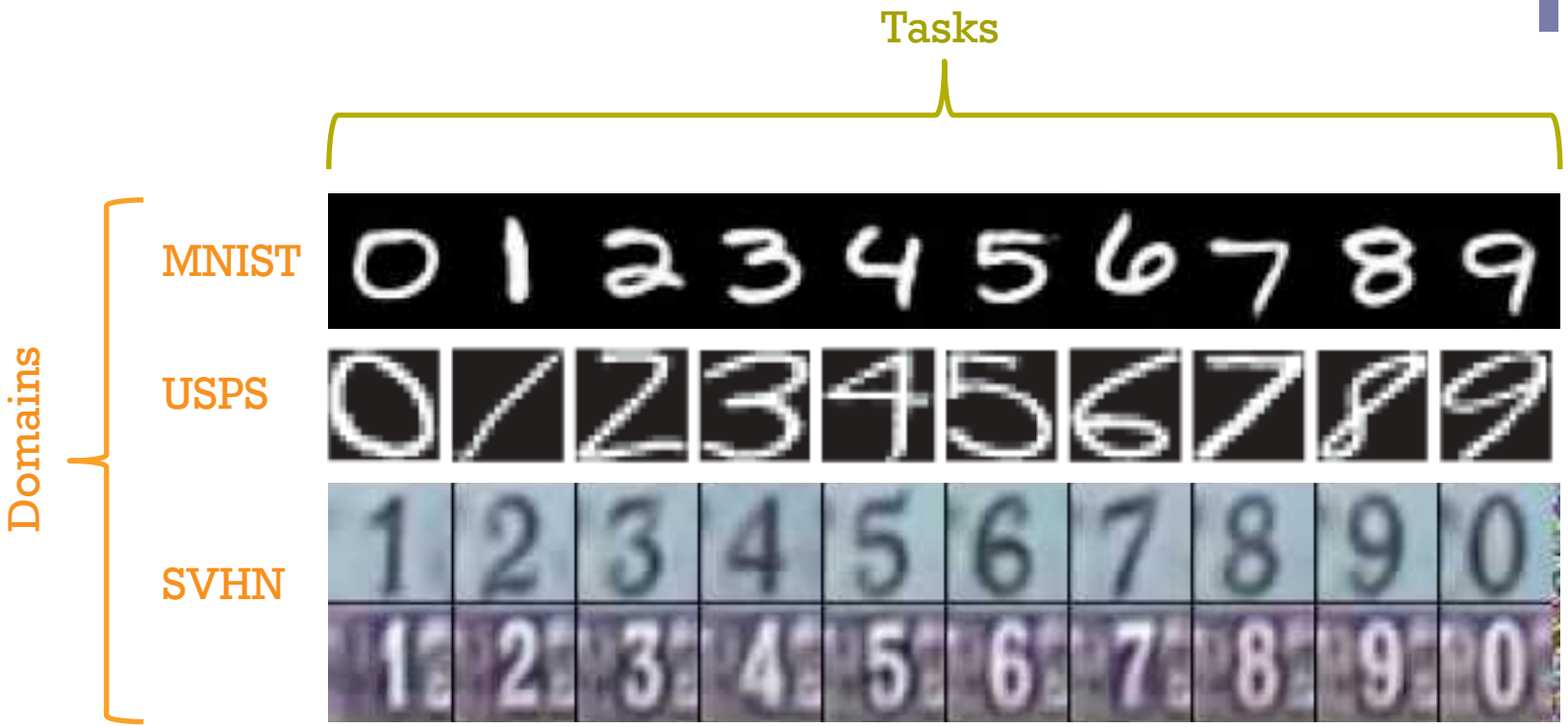
x: Data

z: Task+Domain

$$y = \sigma(P\mathbf{x})\sigma(Q\mathbf{z})^T$$

$$\min_{P,Q} \sum_i \left| y_i - \sigma(P\mathbf{x}_i)\sigma(Q\mathbf{z}_i)^T \right|$$

Multi-Task Multi-Domain: Digits



Simple Way: Concatenate task + domain index: 2-hot task+domain descriptor.
Better ways with tensors….

[Yang & Hospedales, ICLR'15; Wimalawarne, NIPS'14; Romera-paredes, ICML'13]

Related Problem Settings: Summary

Improve Performance

New Settings

Multi-Task → Multi-Task → Zeroshot Recognition

1-hot, Atomic domains/tasks

Distributed domains/tasks

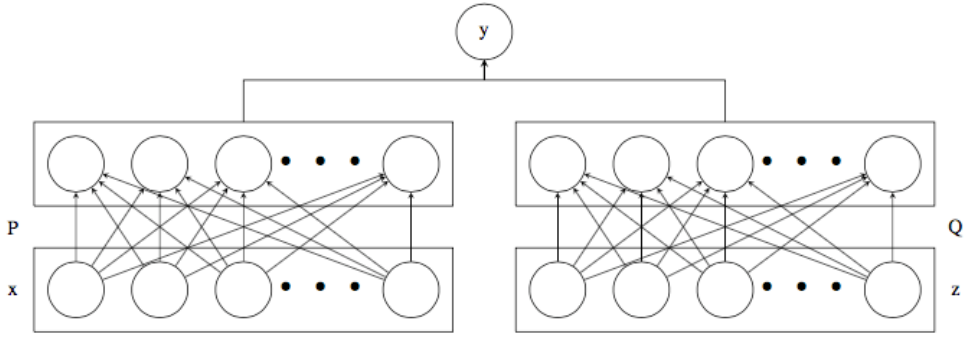
Generalisation Across task/domain descriptions

Multi-Domain → Multi-Domain → Zeroshot Domain Adaptation

Simultaneously Transfer Across Tasks + Domains

MT-MDL

Improve Performance



Overview

- A review of some classic methods
- A general framework
- Example problems and settings
- Going deeper
- Open questions

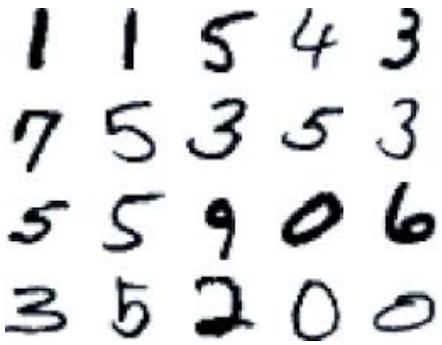
Going Deeper

Outstanding Questions:

- Introduced a NN interpretation of (shallow) MTL.
  - Is there a deep generalisation?
- Looked at MTL/MDL single-output regression/binary classification.
  - What if we want MTL/MDL with multi-output classification/regression?

[Yang & Hospedales, arXiv '16]

Multi-Task Multi-Output

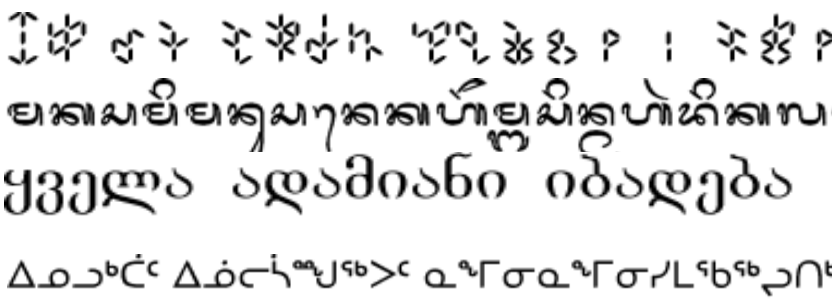


Can share in shallow model

MNIST: Character Recognition:
Tasks: 10x 1-v-all binary tasks?
Tasks: One 10-way multi-class task?

No knowledge sharing in shallow softmax models.
(But outperforms 1-v-All! ☹)
(Can share early layers in Deep model)

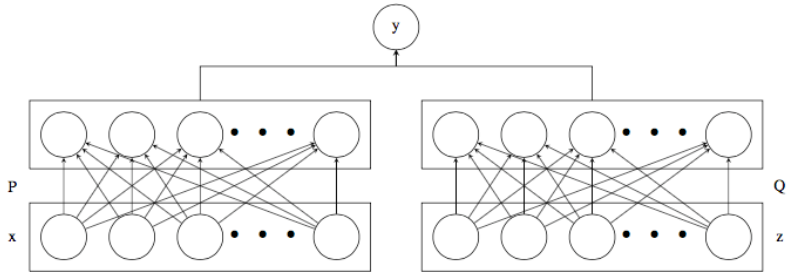
OMNIGLOT: Multi-task Multilingual Character Recognition:
Tasks: 50 languages = 50 tasks.
=> Each task is a multi-class problem



Ideally to share both:
Across classes/chars within each task/language.
Across tasks/languages.

[Yang & Hospedales, arXiv '16]

Multi-Output Multi-Task/Domain



Single-output:
Synthesize a single model weight vector

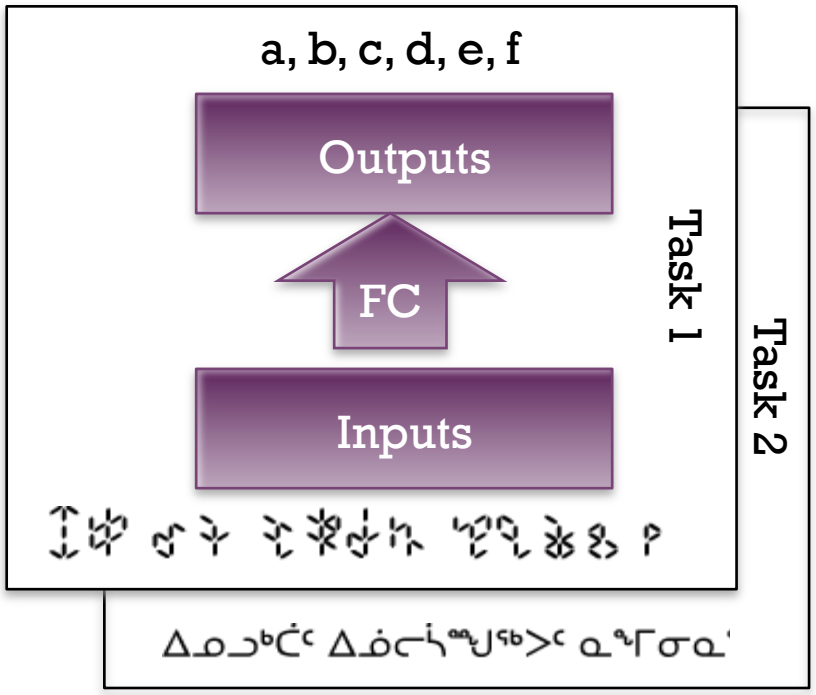
$$y = (\mathbf{w}(\mathbf{z}))^T \mathbf{x} = \mathbf{z}PQ\mathbf{x}$$

Multi-output:
Synthesize a single model weight matrix

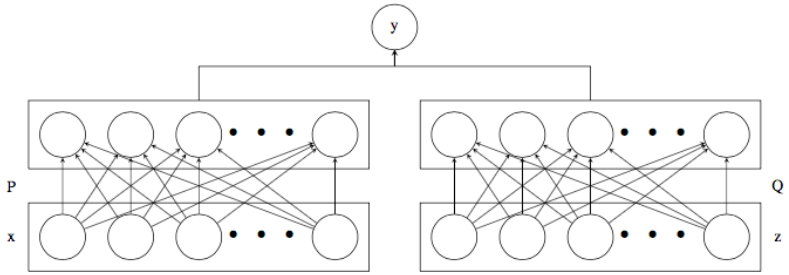
$$\mathbf{y} = \mathbf{W}^T(\mathbf{z})\mathbf{x}$$

Weight generating functions

Multiclass Outputs



Deep Multi-Task Representation Learning

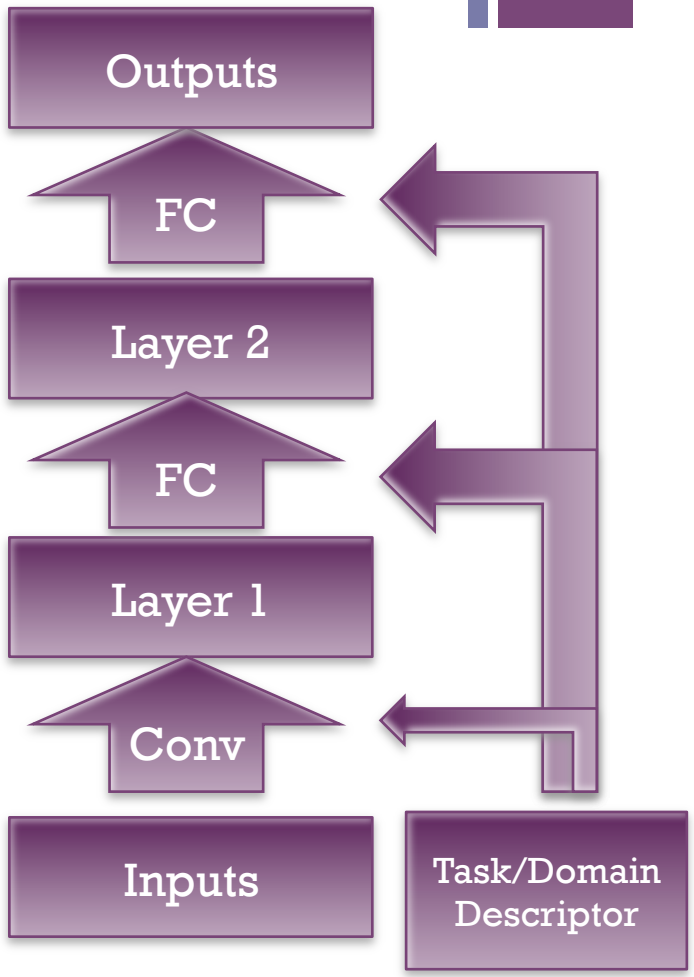


Shallow:
Synthesize a single model weight vector

$$y = (\mathbf{w}(\mathbf{z}))^T \mathbf{x} = \mathbf{z}PQ\mathbf{x}$$

Deep:
Synthesize multiple model weight matrix/tensor:

$$\mathbf{y} = W^T(\mathbf{z})\mathbf{x}$$



[Yang & Hospedales, arXiv '16]

Deep Multi-Task Representation Learning

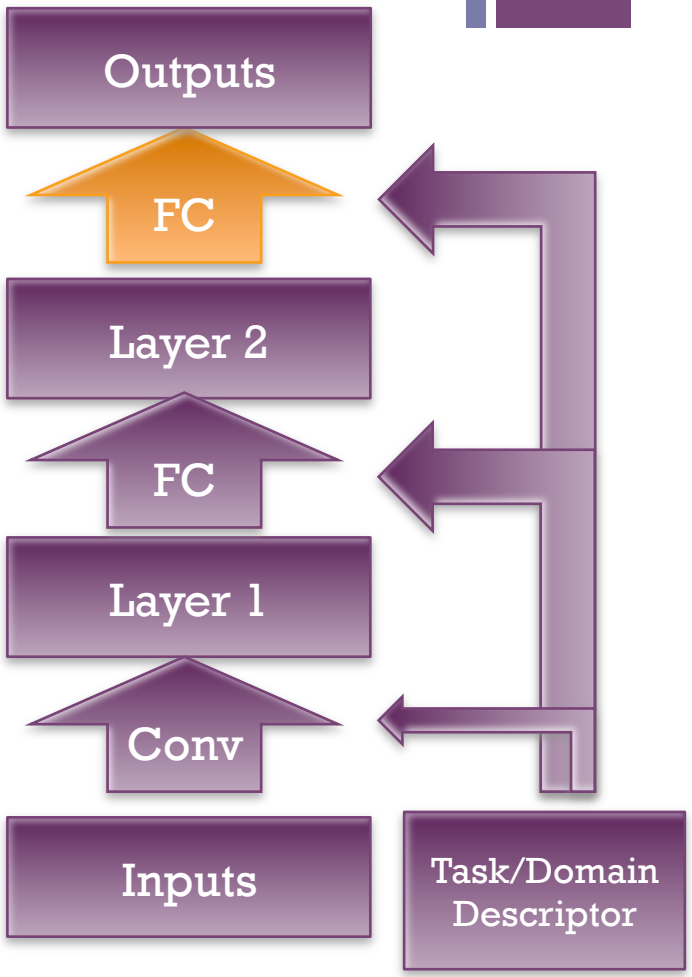
- E.g., One layer of one task needs a 2D matrix:
  - Same layer for all tasks is a 3D tensor.
- Apply tensor "factorization"
  - Recall: Classic MTL as weight matrix factorization

$$y = \mathbf{w}_t^T \mathbf{x} = \left(W\mathbf{z}\right)^T \mathbf{x} = \left(PQ\mathbf{z}\right)^T \mathbf{x}$$

  - "Discriminatively trained" weight tensor factorization

$$\mathcal{W} = \mathcal{S} \bullet U^{(1)} \bullet U^{(2)} \bullet U^{(3)}$$

  - Similarly for conv layers: Higher order tensors
  - Can train end-to-end with backprop.



[Yang & Hospedales, arXiv '16]

Deep Multi-Task Representation Learning

- Classic MTL as weight matrix factorisation

All Task Weights DxT

$$W = PQ$$

Shared Representation DxK

Task-Specific KxT

- Now, weight tensor factorisation

All Task Weights DxTxC

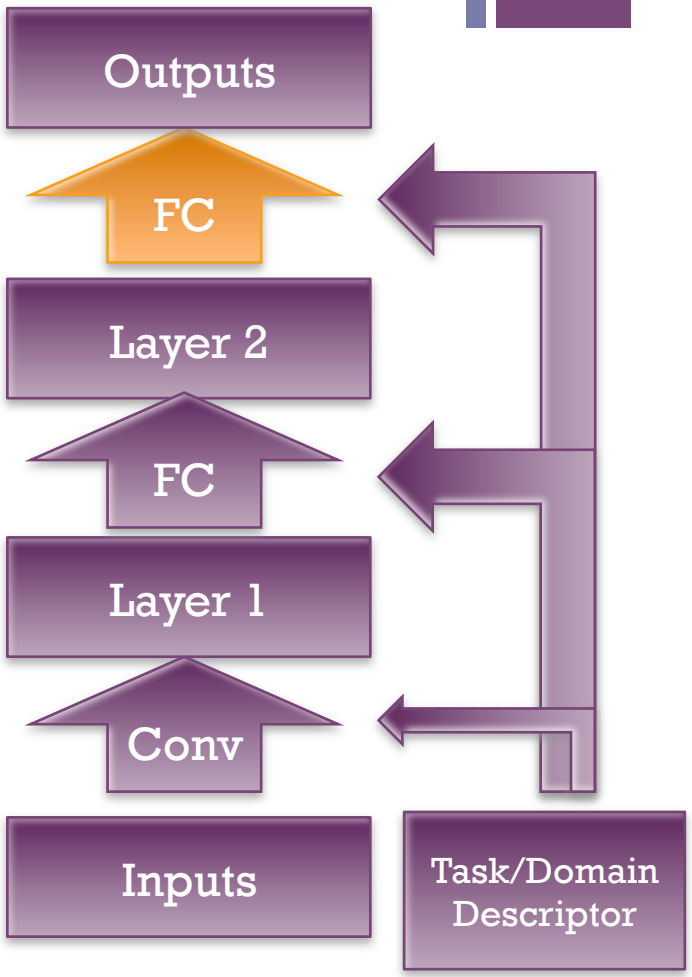
$$\mathcal{W} = S \bullet U^{(1)} \bullet U^{(2)} \bullet U^{(3)}$$

Shared KxKxK

Representation KxD

Task-Specific KxT

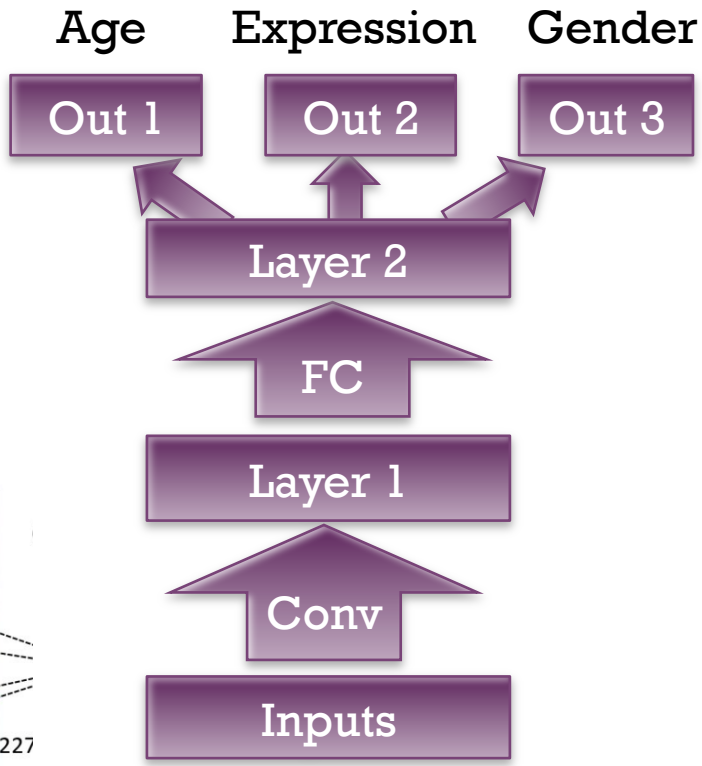
Class-Specific KxC



[Yang & Hospedales, arXiv '16]

Contrast "Deep multi-task"

- In deep learning community, "multi-task" often interpreted as:



- But this is implicitly:

Completely Independent

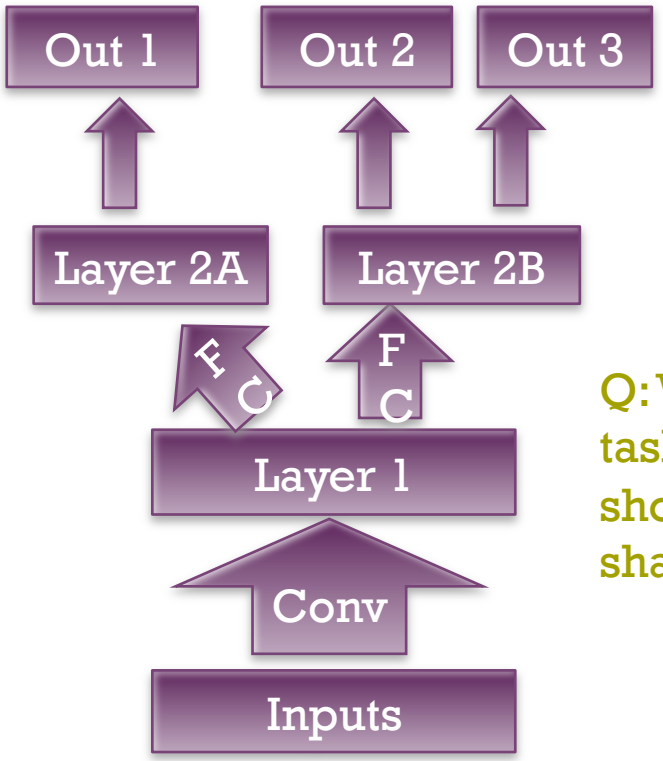
Fully Shared

- i.e., a manually defined sharing structure

E.g., Ranjan, Hyperface, arXiv'16

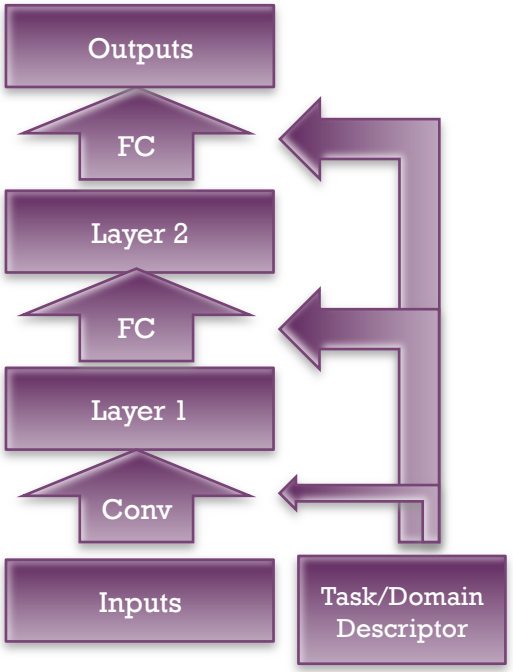
Contrast "Deep multi-task"

- .... But ideal sharing structure is unknown.
  - Depends on (non-uniform) task relatedness.
  - E.g., this may be better:

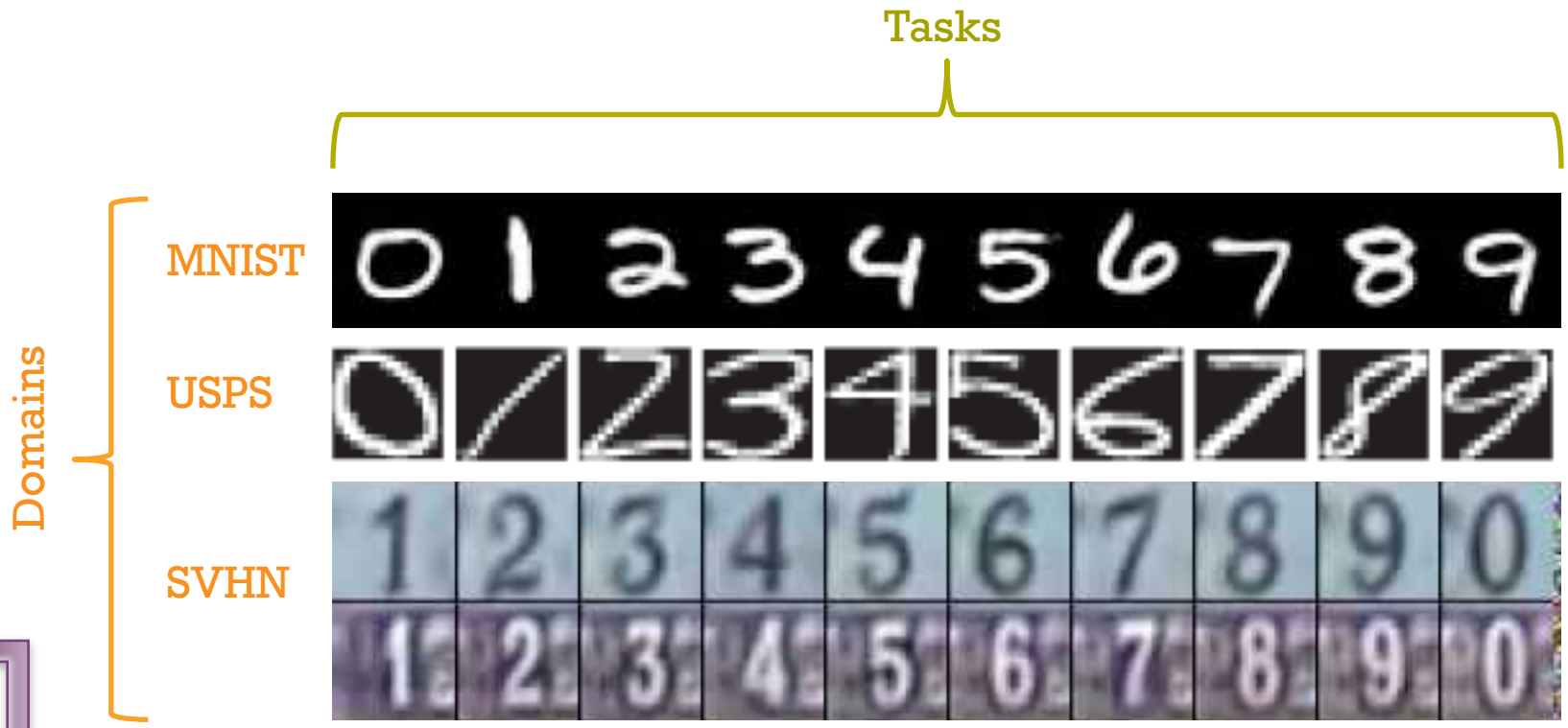


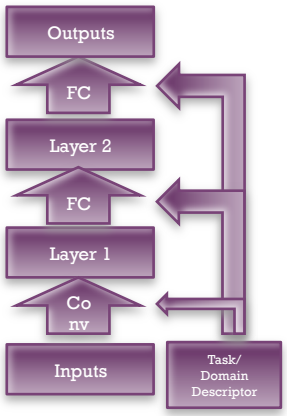
A: Learning the tensor sharing structure at every layer sidesteps explicit architecture search.

Q: Which layers should be task-specific, and which layers should be shared? (And shared between which tasks?)



Deep Multi-(Task/Class) Multi-Domain: Digits





[Yang & Hospedales, ICLR'15; arXiv'16]

Deep Multi-(Task/Class) Multi-Domain: Digits

Tasks

Domains

Amazon

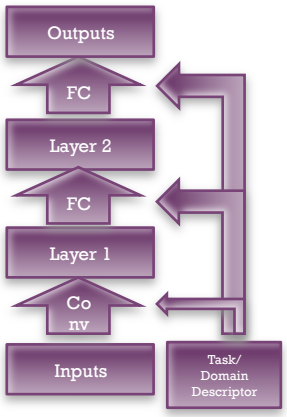
Webcam

DLSR



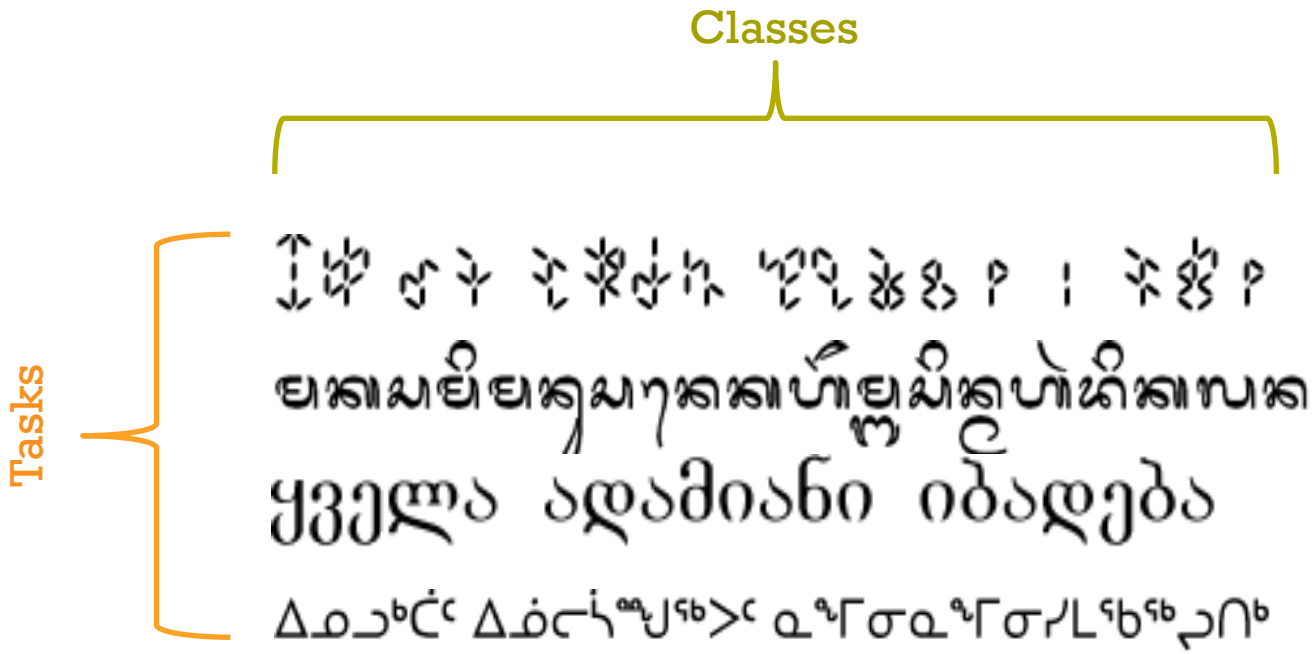


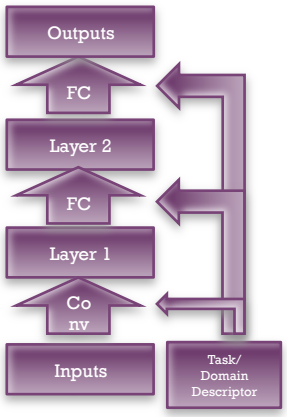




[Yang & Hospedales, arXiv'16]

Deep Multi-Task-Multi-Class: Omniglot



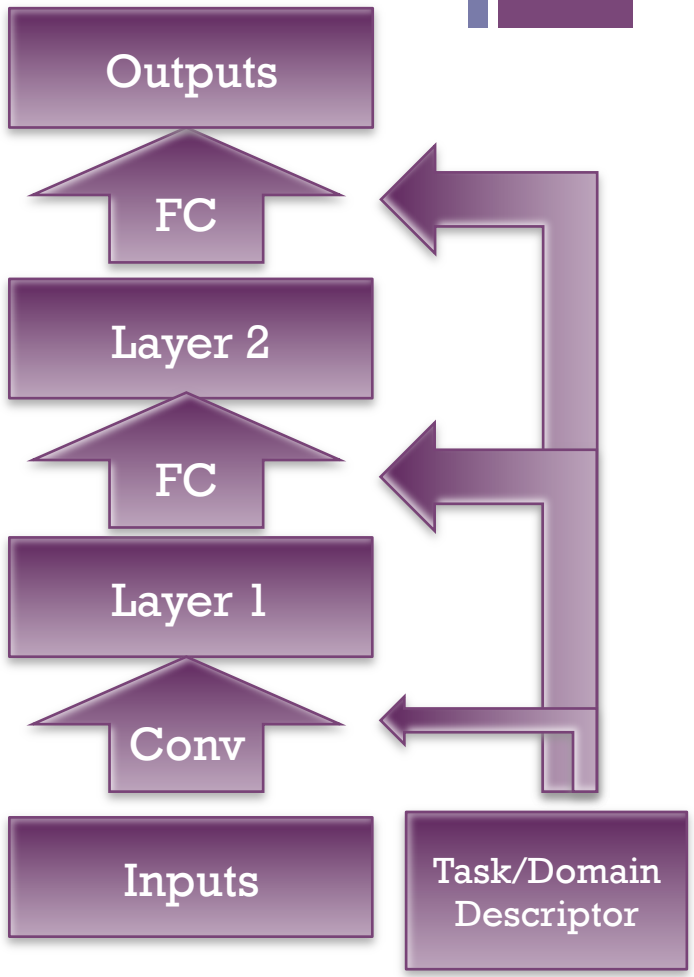


As a byproduct learn how related different languages are (visually) related to each other.

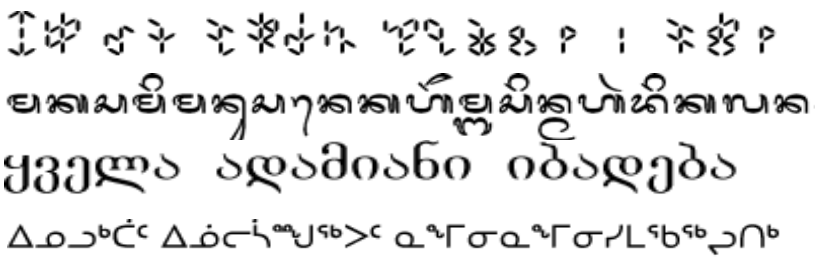
[Yang & Hospedales, arXiv'16]

Deep Multi-Task Representation Learning: Summary

- Generalised the best task subspace-based sharing to deep networks
- Can do both 1-hot and informative tasks/ domains
- Can now solve multi-class/multi-task problems (omniglot) multi-task/multi-domain (office)
- No architecture search



[Yang & Hospedales, arXiv '16]



Overview

- A review of some classic methods
- A general framework
- Example problems and settings
- Going deeper
- Open questions

Open Questions

- Continuous/structured rather than atomic tasks/domains.
- MDL + ZSDA under-studied compared to MTL + ZSL
  - Killer Apps of Zero-Shot Domains?
- Multi-Task/Domain learning with hidden/noisily observed descriptors.
  - Infer descriptors from data => a MTL extension of mixture of experts?
  - Infer current task/domain from reward => Non IID setting.
- Richer abstractions/modularisations for transferring knowledge.
- Life-long learning setting
  - See tasks in sequence. Don't store all the data.
- Speculation: Supervised more interesting than Unsupervised

Thanks For Listening! Any Questions?

- Distributed definitions of task/domains, and different problem settings that arise.
  - MTL => ZSL
  - MDL => ZSDA

- A flexible approach to task/domain transfer
  - Generalizes many existing approaches
  - Covers atomic and distributed task/domain setting, ZSL and ZSDA.
  - Deep extension of shallow MTL/MDL.
  - Sidesteps "Deep-MTL" architecture search

References

- Evgeniou & Pontil, KDD, Regularized multi-task learning, 2004
- Evgeniou et al, JMLR, Learning Multiple Tasks with Kernel Methods, 2005
- Hoffman et al, CVPR, Continuous Manifold Based Adaptation for Evolving Visual Domains, 2014
- Kang et al, ICML, Learning with Whom to Share in Multi-task Feature Learning, 2011
- Khosla, ECCV, Undoing the Damage of Dataset Bias, 2012
- Kumar & Daume, ICML, Learning Task Grouping and Overlap in Multi-task Learning, 2012
- Lampert, CVPR, Predicting the Future Behavior of a Time-Varying Probability Distribution, 2015
- Passos et al, ICML, Flexible Modeling of Latent Task Structures in Multitask Learning, 2012
- Ranjan et al, arXiv:1603.01249, HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition
- Romera-paredes et al, ICML, Multilinear Multitask Learning, 2013
- Salakhutdinov et al, CVPR, Learning to Share Visual Appearance for Multiclass Object Detection, 2011
- Wimalawarne, NIPS, Multitask learning meets tensor factorization: task imputation via convex optimization, 2014
- Yang et al, ACM MM, Cross-domain video concept detection using adaptive SVMs, 2007
- Yang & Hospedales, ICLR, A Unified Perspective on Multi-Domain and Multi-Task Learning, 2015
- Yang & Hospedales, CVPR, Multivariate Regression on the Grassmannian for Predicting Novel Domains, 2016
- Yang & Hospedales, arXiv:1605.06391, Deep Multi-task Representation Learning: A Tensor Factorisation Approach, 2016
- Yang & Hospedales, arXiv, Unifying Multi-Domain Multi-Task Learning: Tensor and Neural Network Perspectives, 2016