On Representation Learning

Alexander G. Ororbia II

Introduction to Machine Learning

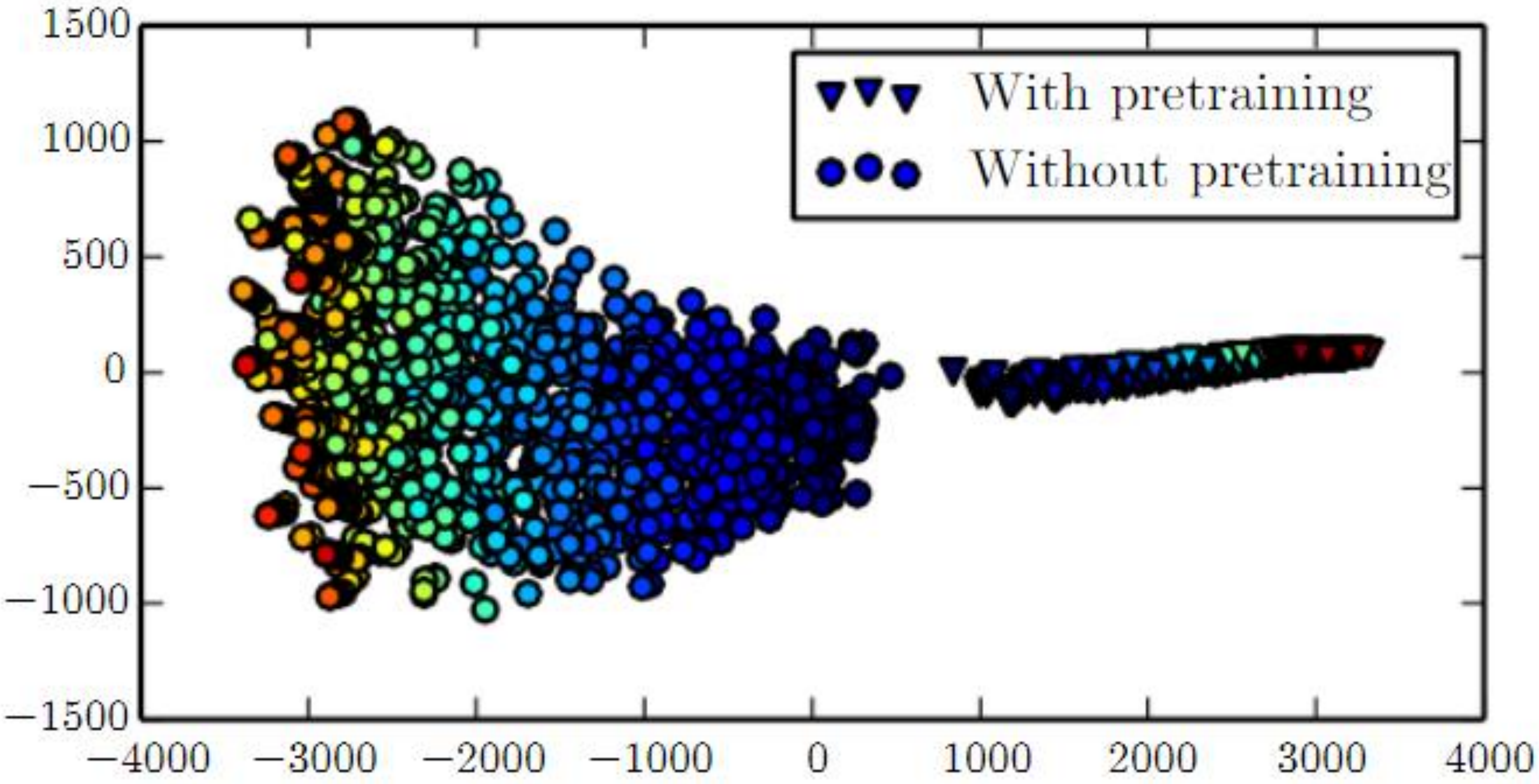
CSCI-736

2/2/2023

Companion reading:
Chapter 15 of Deep Learning textbook

The Problem of Representation Learning

- Representation learning problems face trade-off between preserving as much information about input (as possible) and attaining nice properties, i.e., independence of detectors
  - Models (supervised or unsupervised) have main training objective but learn a representation as a "side effect"
- Often add constraints to shape representation in some wa
  - Density estimation – encourage elements of representation/latent vector **z** to be independent (distributions w/ more independences are easier to model)
- Offers a pathway to facilitate semi-supervised learning
  - ***Hypothesis***: unlabeled data can be used to learn a good representation



With pretraining
Without pretraining
1500
1000
500
0
−500
−1000
−1500
−4000
−3000
−2000
−1000
0
1000
2000
3000
4000

Greedy, Layer-wise Pre-Training

- Learning framework that relies on a single-layer representation learning algorithm (e.g., RBM, single-layer autoencoder, a sparse coding model, etc.)
  - Each layer pretrained via unsupervised learning, taking output of previous layer and producing as output a new representation of data
  - Output has distribution (or relation to other variables, such as categories to predict) that is hopefully "simpler"

- Old idea that dates back as far as the neocognitron (Fukushima, 1975)

On Greedy, Layer-wise Unsupervised Pre-Training

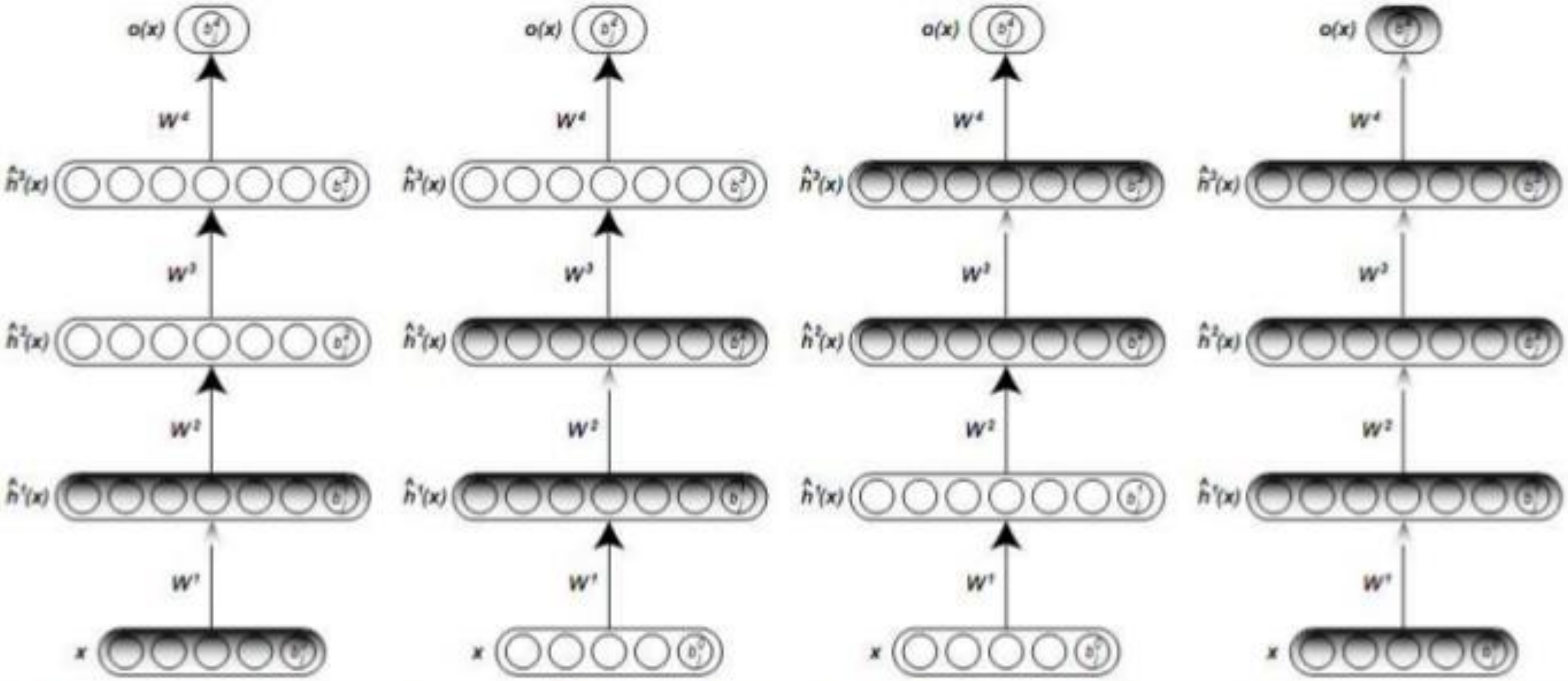
- ***Greedy***
  - Greedy algorithm that optimizes each piece of solution independently (one piece at a time) rather than jointly optimizing all pieces
- ***Layer-wise***
  - Independent pieces are layers of a network
  - Pretraining proceeds one layer at a time, training *k*-th layer while keeping previous ones fixed
  - Lower layers (trained first) are not adapted after upper layers are added
- ***Unsupervised*** = no labels/targets used (not discriminative)
- ***Pre-training*** = a first step before a joint training algorithm is applied (for fine-tuning all layers together)
  - Often viewed as either an intelligent initialization or a regularizer

Algorithm 15.1 *Greedy layer-wise unsupervised pretraining protocol*

Given the following: Unsupervised feature learning algorithm $\mathcal{L}$, which takes a training set of examples and returns an encoder or feature function f. The raw input data is $\boldsymbol{X}$, with one row per example, and $f^{(1)}(\boldsymbol{X})$ is the output of the first stage encoder on $\boldsymbol{X}$. In the case where fine-tuning is performed, we use a learner $\mathcal{T}$, which takes an initial function f, input examples $\boldsymbol{X}$ (and in the supervised fine-tuning case, associated targets $\boldsymbol{Y}$), and returns a tuned function. The number of stages is m.

$f \leftarrow$ Identity function
$\tilde{\boldsymbol{X}} = \boldsymbol{X}$
for $k = 1, \ldots, m$ **do**
  $f^{(k)} = \mathcal{L}(\tilde{\boldsymbol{X}})$
  $f \leftarrow f^{(k)} \circ f$
  $\tilde{\boldsymbol{X}} \leftarrow f^{(k)}(\tilde{\boldsymbol{X}})$
end for
if *fine-tuning* ***then***
  $f \leftarrow \mathcal{T}(f, \boldsymbol{X}, \boldsymbol{Y})$
end if
Return f

Unsupervised greedy layer-wise training procedure.



(a) First hidden layer pre-training

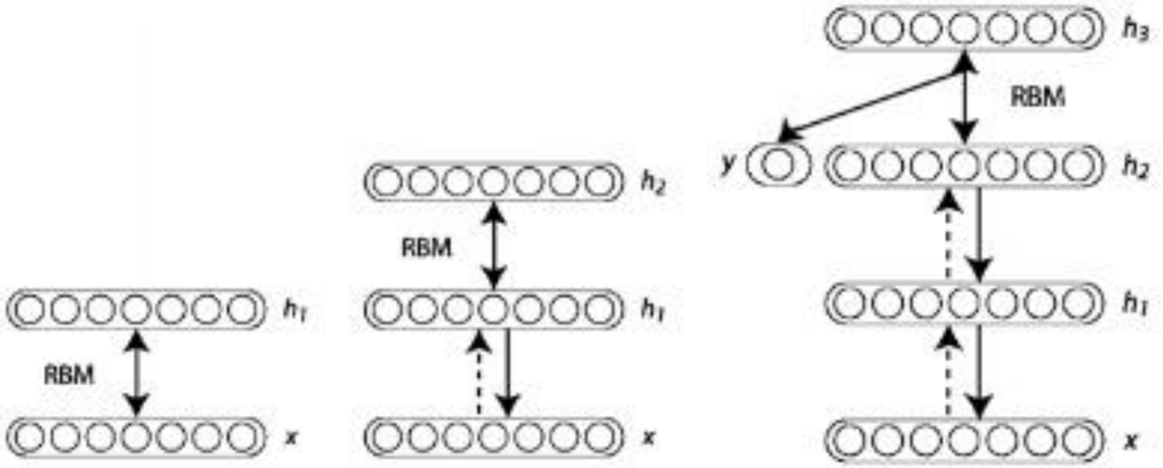
(b) Second hidden layer pre-training

(c) Third hidden layer pre-training

(d) Fine-tuning of whole network

Pretraining: Stacked RBM's

- Iterative pre-training construction of **Deep Belief Network (DBN)** (Hinton et al., 2006)



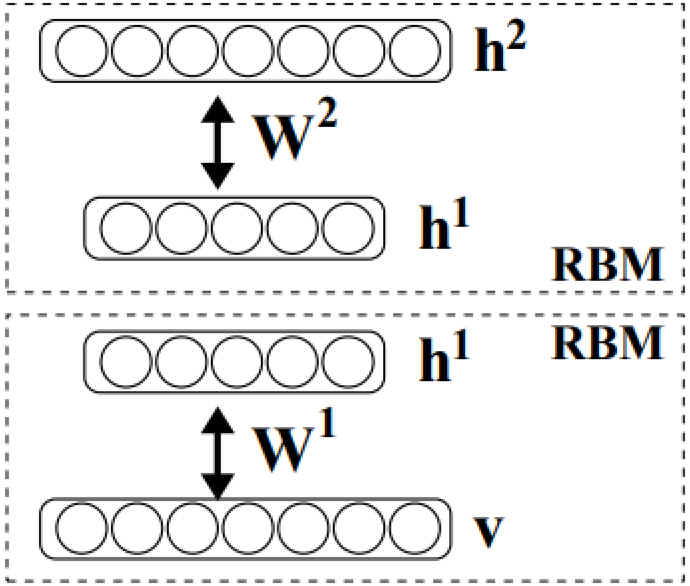
(a) RBM for $\mathbf{x}$ (b) RBM for $\mathbf{h}^1$ (c) RBM for $\mathbf{h}^2$ and y

from: Larochelle et al. (2007). An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation.

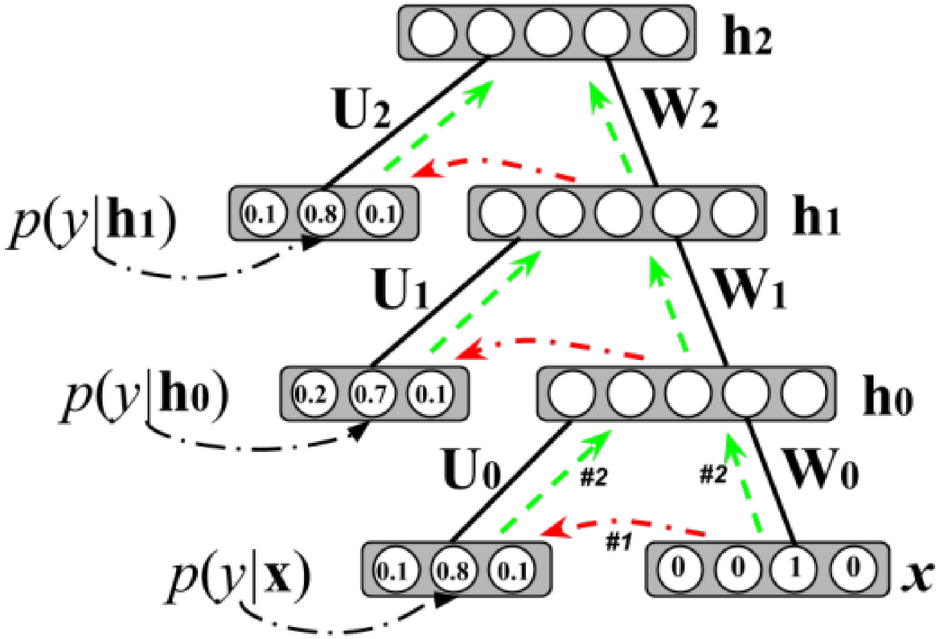
Historical Research Efforts in Pre-Training

Pre-training works! (Erhan et al., 2010), but…

- 2-stage learning (Bengio et al., 2007)
  - Step 1: (Greedy) unsupervised pre-training
    - Deep Belief Networks: Contrastive Divergence (CD-k)
    - Stacked Denoising Autoencoders: Back-propagation w/ cross-entropy loss
  - Step 2: Supervised fine-tuning
    - 1) Toss old model, dump parameters into MLP
    - 2) (Gentle) back-propagation fine-tuning
- Hybrid, single-stage training (Larochelle et al., 2012; Ororbia et al., 2015)
  - Why not learn a generative & discriminative model at same time?



A deep belief network (Salakhutdinov & Murray, 2008)



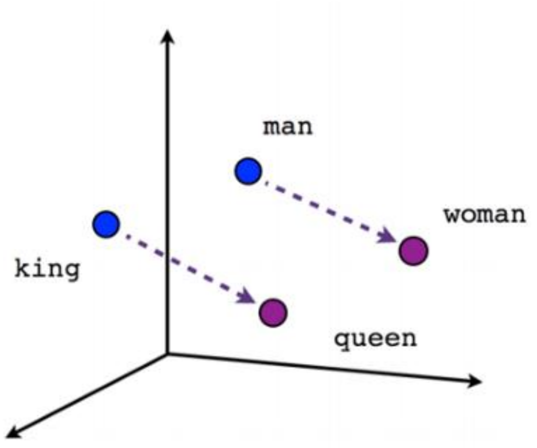
A deep hybrid model (Ororbia et al., 2015)

Why Does Unsupervised Pre-Training Work?

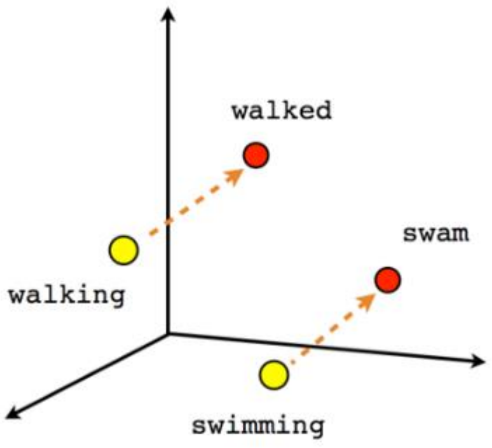
- Greedy layer-wise unsupervised pretraining can yield substantial improvements in test error for classification tasks (sometimes harmful)
- Choice of initial parameters for deep network can have significant regularizing effect on model (and improve optimization)
  - Pretraining initializes the model in inaccessible location?
    - (A region surrounded by areas where cost function varies so much from one example to another that minibatches give a very noisy estimate of gradient)
    - (A region surrounded by areas where Hessian matrix so poorly conditioned that GD methods must use tiny steps)
  - What information gets preserved during supervised fine-tuning?
- Makes use of more general idea that learning about input distribution can help w/ learning about mapping from inputs to outputs
  - Some features useful for unsupervised task also useful for supervised task
  - Generative model of cars/trucks knows about wheels & how many, so supervised learner might be able to access this knowledge

When Might Pre-training Help?

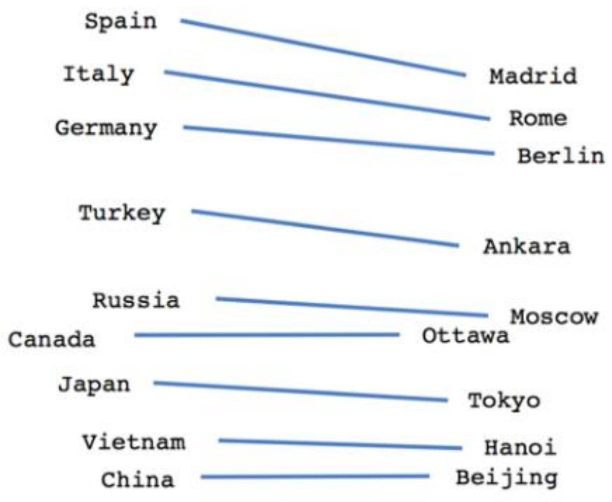
- Unsupervised pretraining to be more effective when initial representation is poor
  - ***Example***: word embeddings (encode similarity between words by distance from each other vs. one-hots which are equally distant from each other)



Male-Female



Verb tense



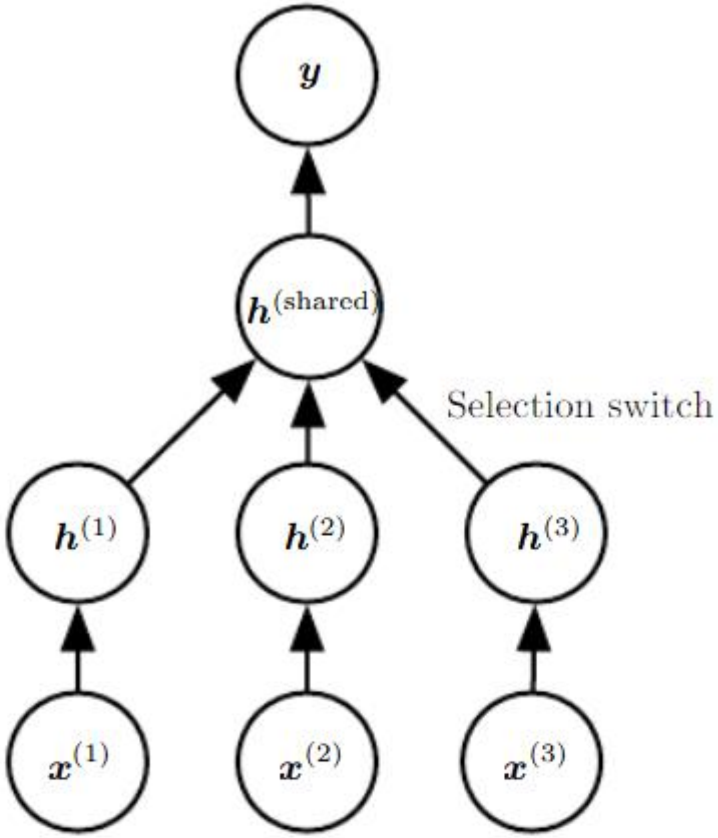
Country-Capital

When Might Pre-training Help?

- When number of labeled examples is small
  - Pretraining might perform best when number of unlabeled examples is very large
- Function to be learned is extremely complicated
  - Pre-training does not bias learner toward discovering a simple function (as in L1/L2 regularization) – leads learner to discovering feature functions useful for unsupervised learning task
  - If true underlying functions are complicated & shaped by regularities of input distribution => unsupervised learning can be more appropriate regularizer

Transfer Learning

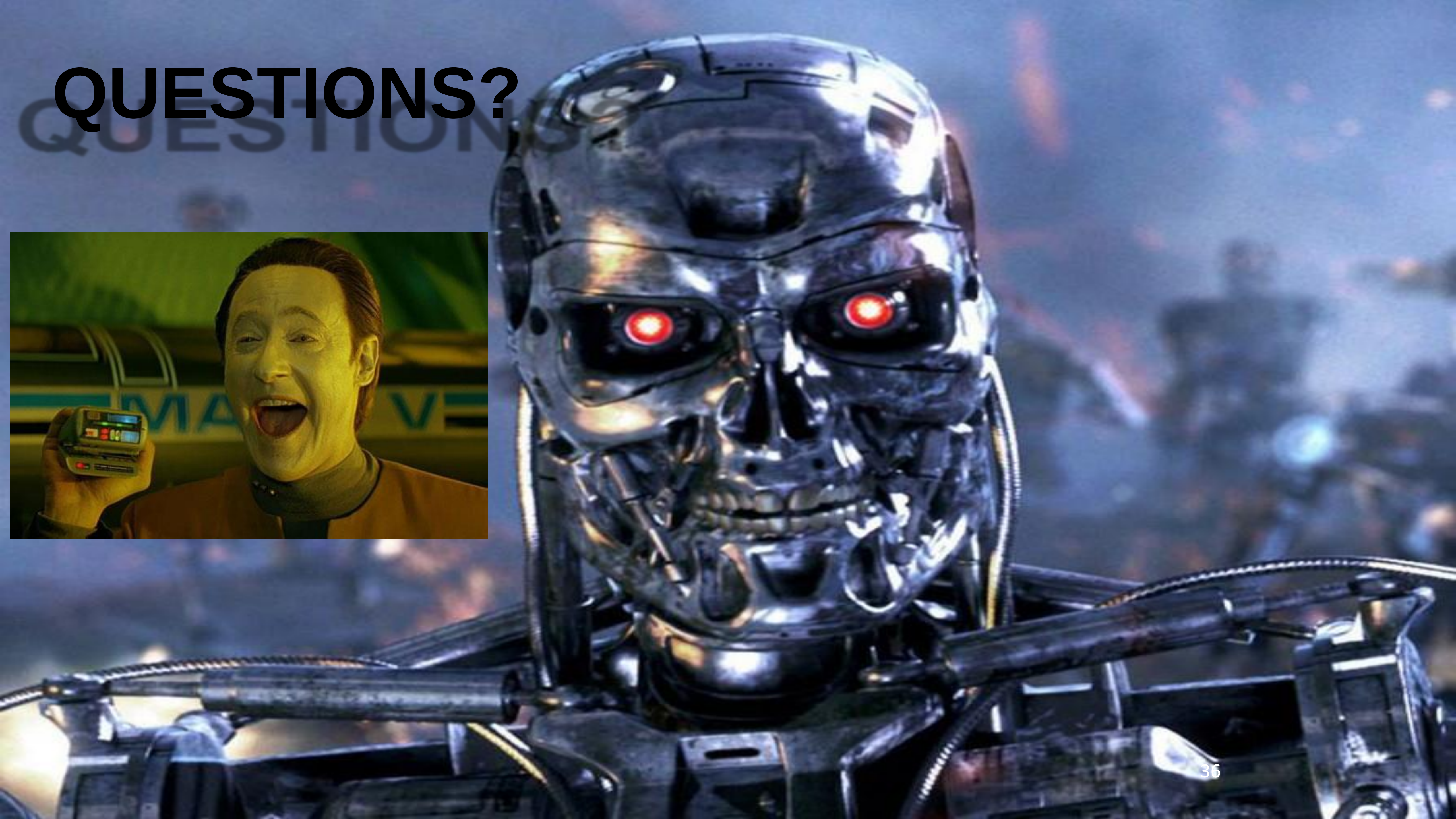
- **Transfer learning**: learner must perform two or more different tasks
  - We assume that many of factors of variation in P1arerelevant to factors of variations needed for learning P2.
  - ***Example***: supervised learning, where input is same but target may be of a different nature (different classes/categories)
- Many visual categories share low-level notions, e.g., edges and visual shapes, effects of geometric changes, changes in lighting
- Representation learning useful when there exist features that are useful for different settings/tasks (corresponding to underlying factors that appear in more than one setting)



Lower levels (up to selection switch) are task-specific, upper levels are shared -- lower levels learn to translate task-specific input into a generic set of features (above case = shared output semantics)

Domain Adaptation

- **Domain adaptation**: task (and optimal input-to-output mapping) remains same between each setting, but the input distribution is slightly different
  - Concept drift = gradual changes in data distribution over time
- Objective: take advantage of data from first setting to extract information that may be useful when learning or predicting in second setting
  - Representation learning can help when same representation is useful in both settings – using same representation in both settings allows representation to benefit from training data available for both tasks
- One-shot learning, zero-shot learning/zero-data learning



QUESTIONS?