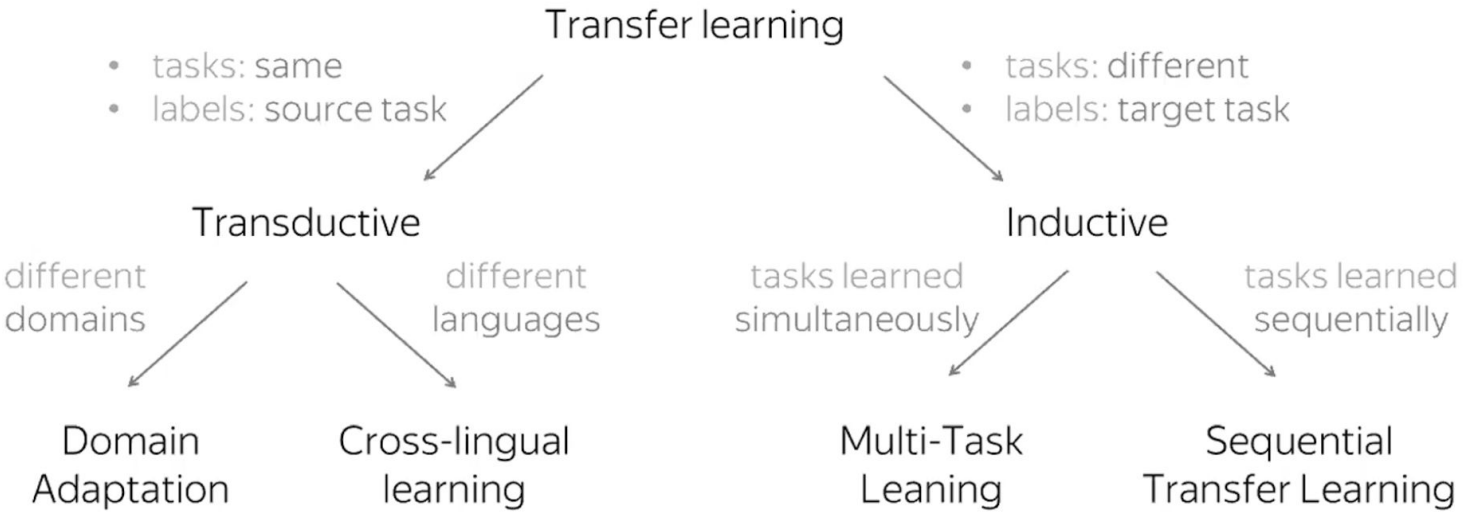
Yandex

Domain Adaptation

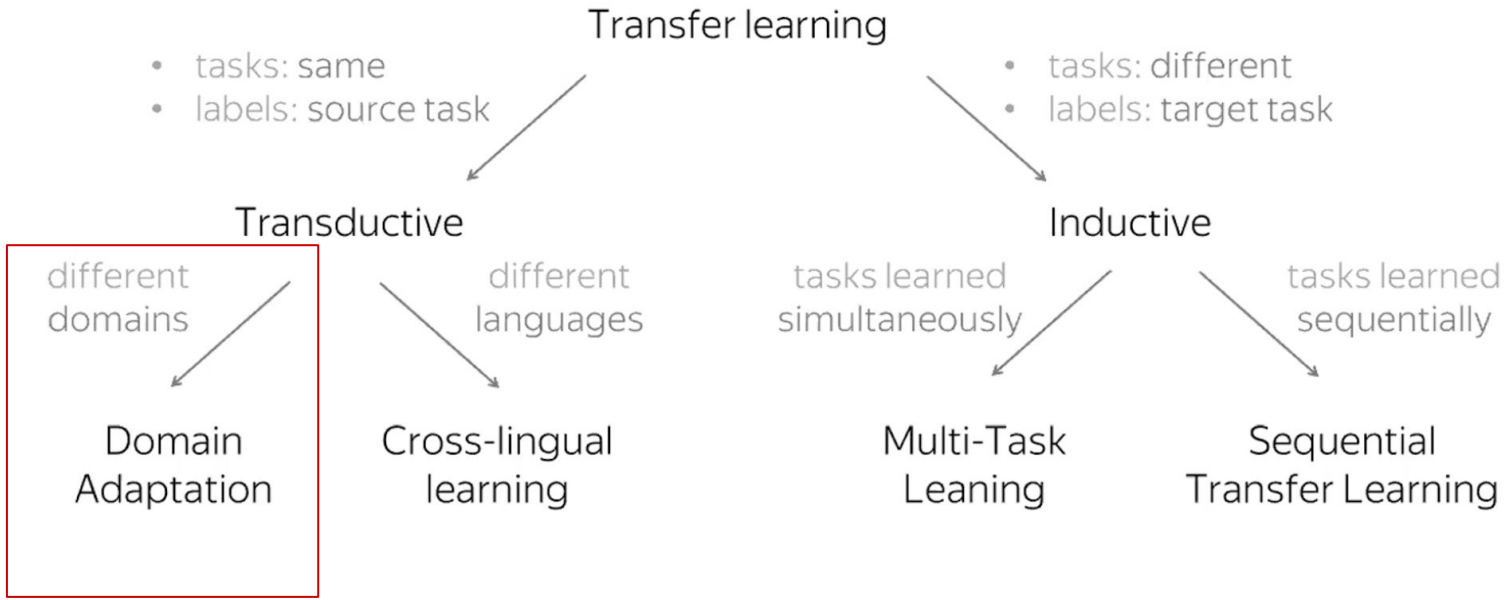
October 15, 2020

A Taxonomy of Transfer Learning in NLP



This taxonomy is from Ruder, 2019

A Taxonomy of Transfer Learning in NLP



This taxonomy is from Ruder, 2019

Standard supervised learning setting

Standard supervised learning setting

- Model (CNN vs RNN, GNMT vs Transformer and so on) $\mathcal{H} = \{h_\theta : X \to Y\}$

Standard supervised learning setting

- Model (CNN vs RNN, GNMT vs Transformer and so on) $\mathcal{H} = \{h_\theta : X \to Y\}$

- Data (source and target marginal distributions)
  source ~ training data
  target ~ testing data

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Standard supervised learning setting

- Model (CNN vs RNN, GNMT vs Transformer and so on) $\mathcal{H} = \{h_\theta : X \to Y\}$

- Data (source and target marginal distributions)
  source ~ training data
  target ~ testing data

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

- Task (loss function, risk) $h_\theta^* = \arg\min_\theta R^S(h_\theta) = \arg\min_\theta \mathrm{E}_{(x,y)\sim P_S}[L(h_\theta(x), y)]$

Standard learning assumptions

Standard learning assumptions

- Sample points are IID

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \overset{iid}{\sim} P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \overset{iid}{\sim} P_T(x, y)$$

Standard learning assumptions

- Sample points are IID

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \overset{iid}{\sim} P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \overset{iid}{\sim} P_T(x, y)$$

- Same distributions for training and test

$$P_S(x, y) = P_T(x, y)$$

Standard learning assumptions

- Sample points are IID

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \overset{iid}{\sim} P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \overset{iid}{\sim} P_T(x, y)$$

- Same distributions for training and test

$$P_S(x, y) = P_T(x, y)$$

- Fixed distributions (they don’t change with time)

Standard learning assumptions

- Sample points are IID

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \overset{iid}{\sim} P_S(x,y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \overset{iid}{\sim} P_T(x,y)$$

- Same distributions for training and test

$$P_S(x,y) = P_T(x,y)$$

- Fixed distributions (they don't change with time)

$$\epsilon_T \leq \hat{\epsilon}_S + O(\frac{complexity(h)}{\sqrt{N}})$$

$$\epsilon_T = \mathop{\mathrm{E}}_{(x,y)\sim P_T}[L(h_\theta(x), y)]$$

$$\hat{\epsilon}_S = \frac{1}{N}\sum_{i=1}^{N} L(h_\theta(x^S), y)$$

Standard learning assumptions

- Sample points are IID

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \overset{iid}{\sim} P_S(x,y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \overset{iid}{\sim} P_T(x,y)$$

- Same distributions for training and test

$$P_S(x,y) = P_T(x,y)$$

EXPECTATIONS

- Fixed distributions (they don't change with time)

Real-world problems

- Training sample is biased
- Noisy labels for training sample
- Sample points are not drawn IID
- Distributions drifts with time

REALITY

Domain shift:

$$P_S(x, y) \neq P_T(x, y)$$

Real-world examples

- Data from different sources
- Absence of good in-domain data
- Temporal (both short term and long term) changes of audience structure
- Synthetic training data

Real-world examples

- Data from different sources
- Absence of good in-domain data
- Temporal (both short term and long term) changes of audience structure
- Synthetic training data

Domain shift:

$$P_S(x, y) \neq P_T(x, y)$$

(This slide is based on wonderful The Future of Natural Language Processing lecture)

In-domain vs. out-of-domain generalization

On SQuAD Dani Yogatama et al., "Learning and Evaluating General Linguistic Intelligence," http://arxiv.org/abs/1901.11373

	SQuAD	Trivia	QuAC	QA-SRL	QA-ZRE
BERT	86.5 (78.5)	35.6 (13.4)	56.2 (43.9)	77.5 (65.0)	55.3 (40.0)
ELMo	81.8 (72.2)	32.9 (12.6)	45.0 (34.5)	68.7 (52.3)	60.2 (42.0)

Table 2: F_1 (exact match) scores of the best BERT and ELMo models trained on SQuAD and evaluated on other question answering datasets.

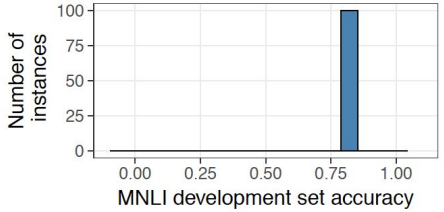


Figure 2: In-distribution generalization: Performance on the MNLI development set

On MNLI: R. Thomas McCoy, Junghyun Min, and Tal Linzen, "BERTs of a Feather Do Not Generalize Together: Large Variability in Generalization across Models with Similar Test Set Performance," http://arxiv.org/abs/1911.02969

Heuristic	Definition	Example
Lexical overlap	Assume that a premise entails all hypotheses constructed from words in the premise	**The doctor** was **paid** by **the actor**. $\xrightarrow[\text{WRONG}]{}$ The doctor paid the actor.
Subsequence	Assume that a premise entails all of its contiguous subsequences.	The doctor near **the actor danced**. $\xrightarrow[\text{WRONG}]{}$ The actor danced.
Constituent	Assume that a premise entails all complete subtrees in its parse tree.	If **the artist slept**, the actor ran. $\xrightarrow[\text{WRONG}]{}$ The artist slept.

Figure 1: The heuristics targeted by the HANS dataset, along with examples of incorrect entailment predictions that these heuristics would lead to. (Figure from McCoy et al. (2019).)

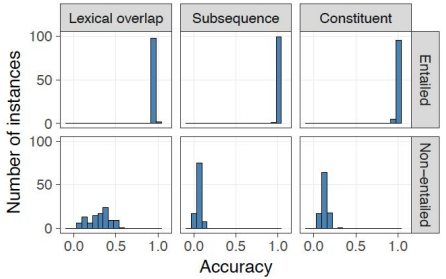


Figure 3: Out-of-distribution generalization: Performance on the HANS evaluation set, broken down into six categories of examples based on which syntactic heuristic each example targets and whether the correct label is *entailment* or *non-entailment*. The non-entailed lexical overlap cases (lower left plot) display a large degree of variability across instances.

(This slide is also based on wonderful The Future of Natural Language Processing lecture)

In-domain vs. out-of-domain generalization

On MNLI: R. Thomas McCoy, Junghyun Min, and Tal Linzen, "BERTs of a Feather Do Not Generalize Together: Large Variability in Generalization across Models with Similar Test Set Performance," http://arxiv.org/abs/1911.02969

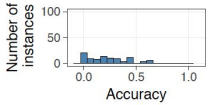
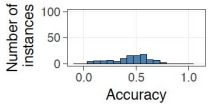
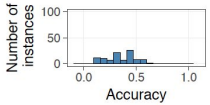
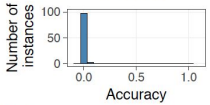
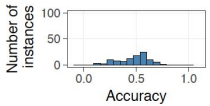
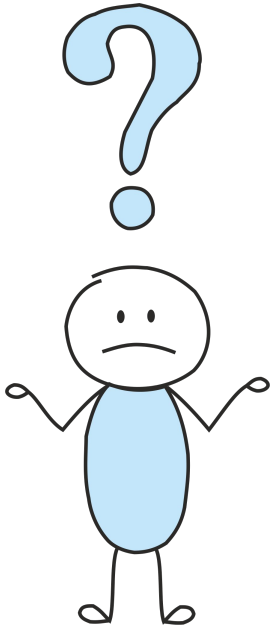
Category	Example	Accuracy distribution
Subject-object swap	The doctor visited the lawyer. ↛ The lawyer visited the doctor.	
Preposition	The judges by the manager saw the artists. ↛ The artists saw the manager.	
Relative clause	The actors advised the author who the tourists saw. ↛ The author advised the tourists.	
Passive	The senators were recommended by the managers. ↛ The senators recommended the managers.	
Conjunction	The doctors advised the presidents and the tourists. ↛ The presidents advised the tourists.	

Figure 4: Results on the various subcategories within the non-entailed lexical overlap examples of the HANS dataset. We do not include the other 25 subcategories of the HANS dataset in this figure as there was little variability across instances for those subcategories.

How to check whether there is a domain shift or not?



How to check whether there is a domain shift or not?

If domain distribution are different we can train classifier to discriminate them.

$$P_S(x, y) \neq P_T(x, y)$$

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

$$P_S(x, y) \neq P_T(x, y)$$

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\boxed{\epsilon_T} \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Expected target error

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Expected source error

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

The divergence between source and target domain

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{(h,h')} \left| \mathop{\mathrm{E}}_{(x,y)\sim P_S} [h(x) \neq h'(x)] - \mathop{\mathrm{E}}_{(x,y)\sim P_T} [h(x) \neq h'(x)] \right|$$

H-divergence measures the worst case of the disagreement between a pair of hypothesis.
How much features discriminative for S and T

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

The divergence between source and target domain

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) \sim d_{\mathcal{A}} = 2(1 - 2\varepsilon)$$

As H-divergence approximation proxy A-distance (PAD) can be used.
ε is the generalization error of the domain classifier.

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Error of ideal joint hypothesis

$$\lambda = \min_h [\epsilon_S(h) + \epsilon_S(h)]$$

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Generalization error under domain shift

Ben-David, 2010

Ben-David theory:

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Domain adaptation tend to minimize second and (third?) terms.

Domain adaptation

- Unsupervised DA:
  No labels for target domain

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{T} = \{x_i^T\}_{i=1}^M \sim P_T(x)$$

- (Semi)supervised DA:
  Target domain dataset is (partially) labeled

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Covariate shift:

$$P_S(x) \neq P_T(x)$$

$$P_S(y|x) = P_T(y|x)$$

Why does a covariate shift make problems?

$$P_S(x) \neq P_T(x)$$

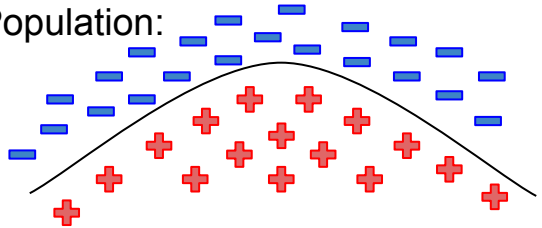
$$P_S(y|x) = P_T(y|x)$$

Why does a covariate shift make problems?

$$P_S(x) \neq P_T(x)$$

$$P_S(y|x) = P_T(y|x)$$

Population:

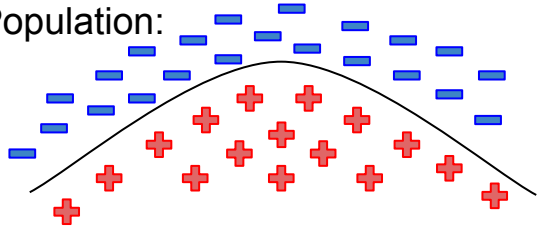


Why does a covariate shift make problems?

$$P_S(x) \neq P_T(x)$$

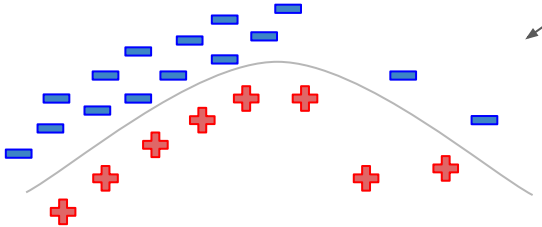
$$P_S(y|x) = P_T(y|x)$$

Population:



Source distribution:

Biased sampling

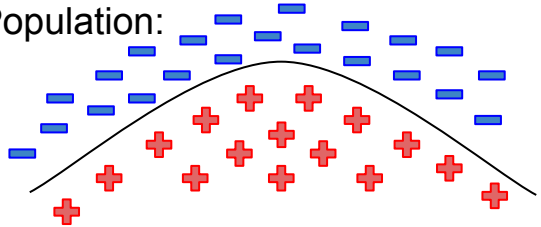


Why does a covariate shift make problems?

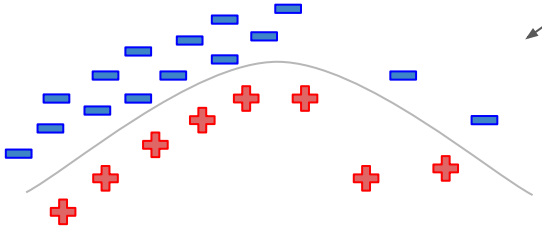
$$P_S(x) \neq P_T(x)$$

$$P_S(y|x) = P_T(y|x)$$

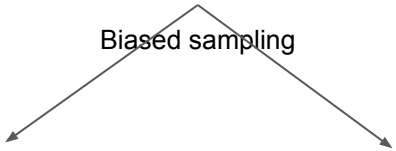
Population:



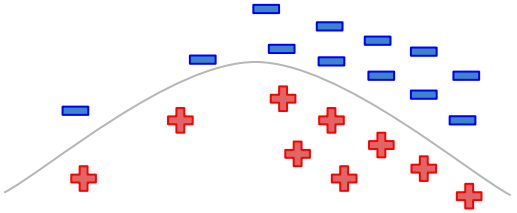
Source distribution:



Biased sampling



Target distribution:

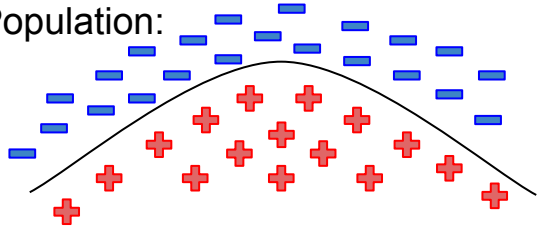


Why does a covariate shift make problems?

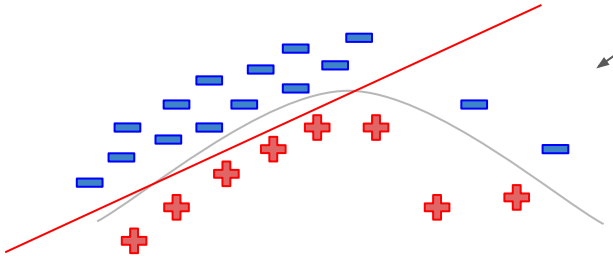
$$P_S(x) \neq P_T(x)$$

$$P_S(y|x) = P_T(y|x)$$

Population:

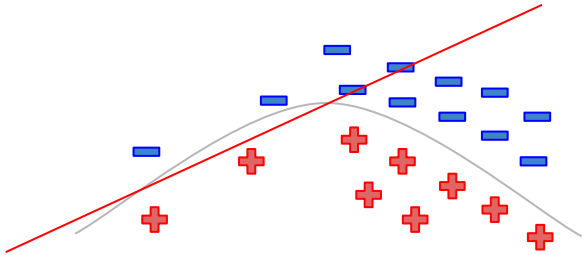


Source distribution:



Biased sampling

Target distribution:

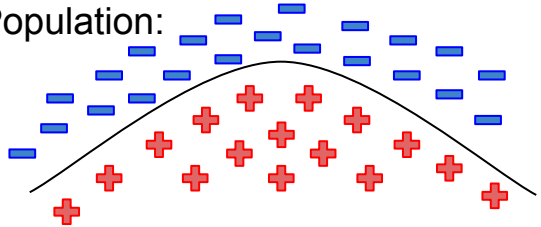


Why does a covariate shift make problems?

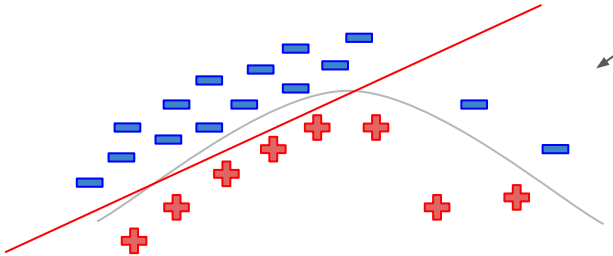
$$P_S(x) \neq P_T(x)$$

$$P_S(y|x) = P_T(y|x)$$

Population:

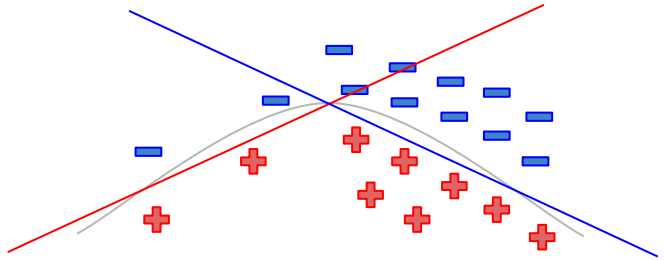


Source distribution:



Biased sampling

Target distribution:



Not optimal decision rule!

Unsupervised domain adaptation

Unsupervised domain adaptation

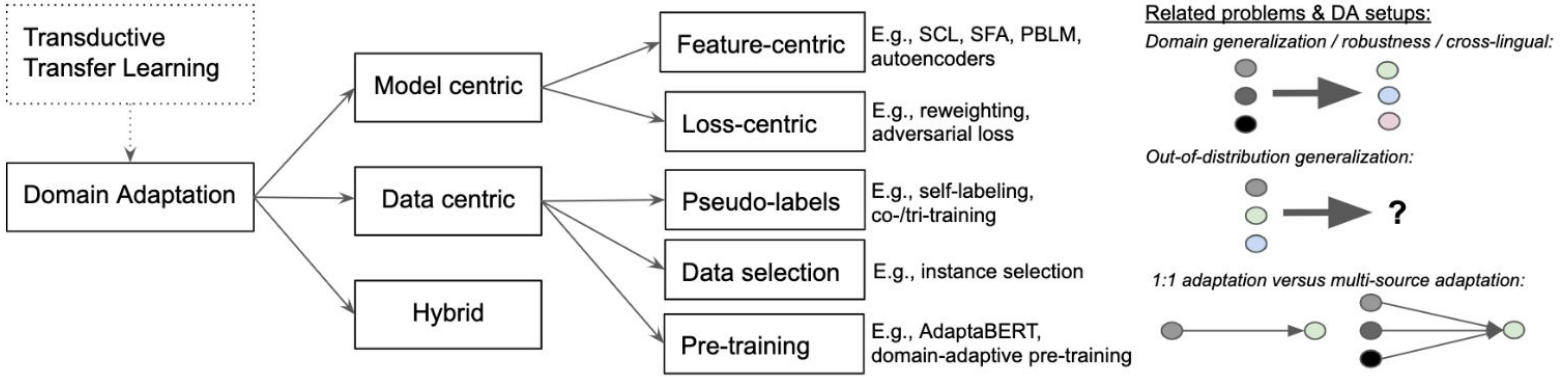


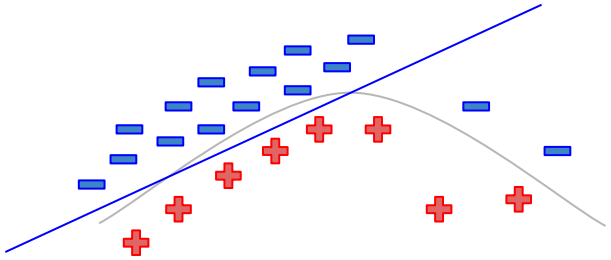
Figure 1: Taxonomy of DA as special case of transductive transfer learning, DA setups (1:1 versus multi-source adaptation) and related problems (domain and out-of-distribution generalization).

[Neural Unsupervised Domain Adaptation in NLP—A Survey]

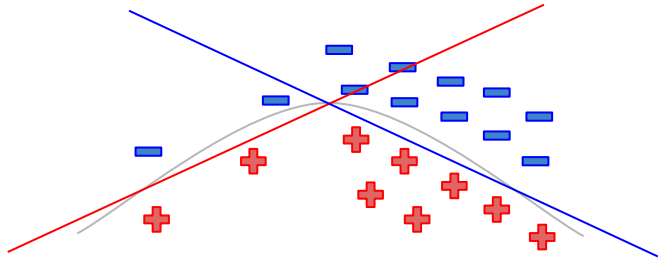
Instance weighting

$$h_\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L(h_\theta(x), y)$$

Source distribution:



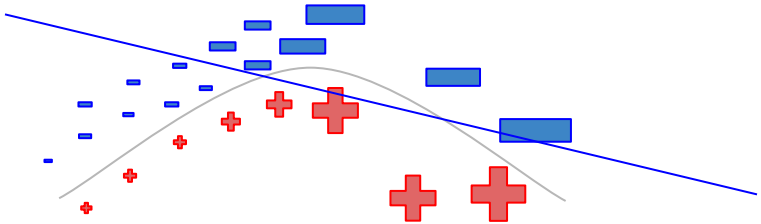
Target distribution:



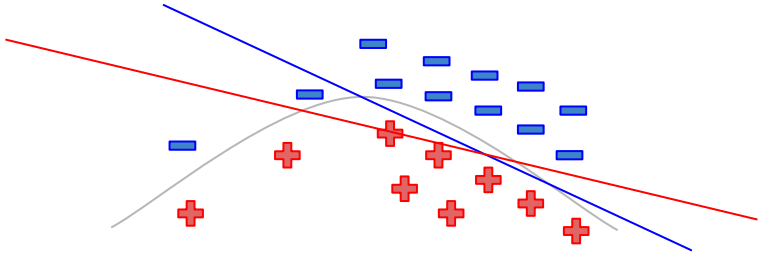
Not optimal decision rule!

Instance weighting

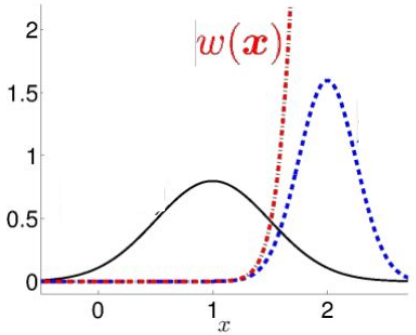
Source distribution:



Target distribution:



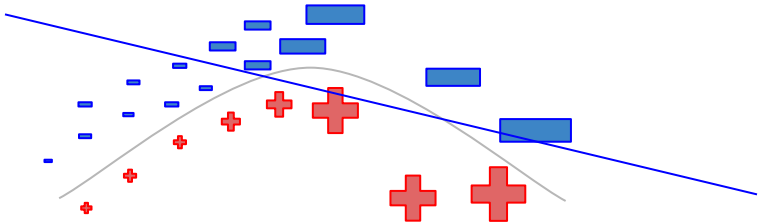
Instance weighting



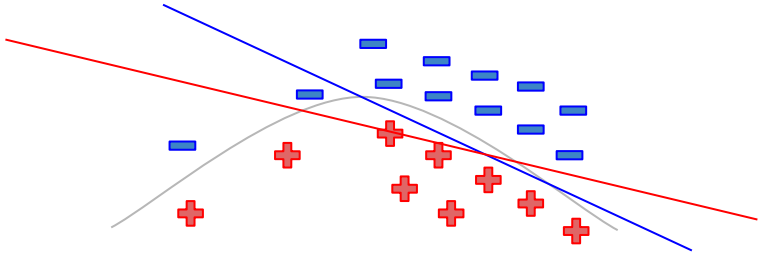
$$h_\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} w(x_i) L(h_\theta(x_i), y_i)$$

$$w(x) = \frac{p_T(x)}{p_S(x)}$$

Source distribution:



Target distribution:



Instance weighting: NLP applications

- Neural Machine Translation

  $x = (v_1, v_2, ..., v_k)$: Input sequence of tokens

  $P_T(x), P_S(x) = ?$

Instance weighting: NLP applications

- Neural Machine Translation

  $x = (v_1, v_2, ..., v_k)$: Input sequence of tokens

  $P_T(x), P_S(x)$ can be defined using source and target language models

Instance weighting: NLP applications

- Neural Machine Translation

  $x = (v_1, v_2, ..., v_k)$: Input sequence of tokens

  $P_T(x), P_S(x)$ can be defined using source and target language models

  Weighting using language models (Wang et al. 2017):

  $$w_i = \delta[H_S(x) - H_T(x)]$$

  $$H = -\frac{1}{k} \log P(x)$$

  (δ - min-max normalization)

Instance weighting: NLP applications

- Neural Machine Translation

  $x = (v_1, v_2, ..., v_k)$: Input sequence of tokens

  $P_T(x), P_S(x)$ can be defined using source and target language models

  Weighting using domain classifier (Chen et al. 2017):

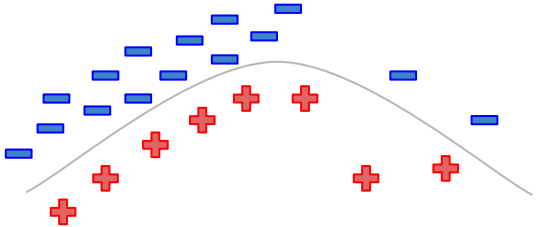
  $$w_i = (1 + p_d(x_i))$$

  $p_d(x)$ - probability of being from target

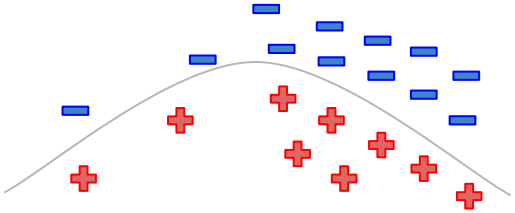
Data selection

Instead of weighting, we can train classifier and drop all observations that are too dissimilar from target domain.

Source distribution:



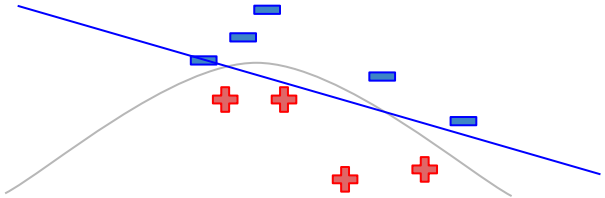
Target distribution:



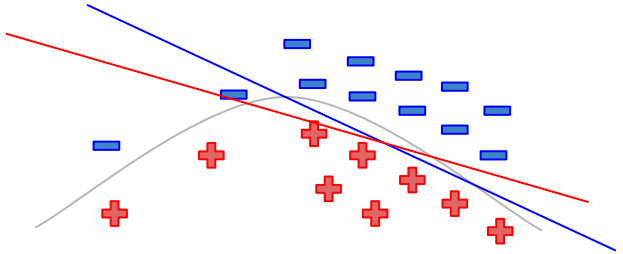
Data selection

Instead of weighting, we can train classifier and drop all observations that are too dissimilar from target domain.

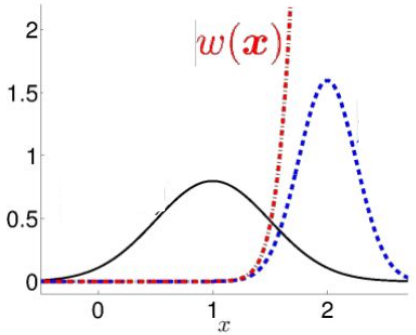
Source distribution:



Target distribution:



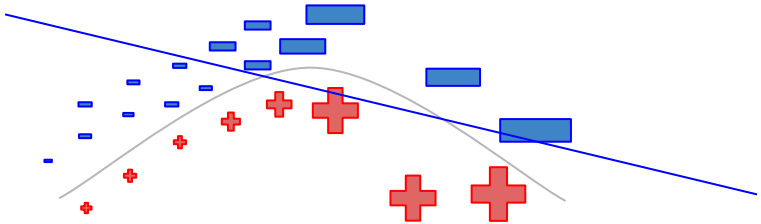
Instance weighting: problems



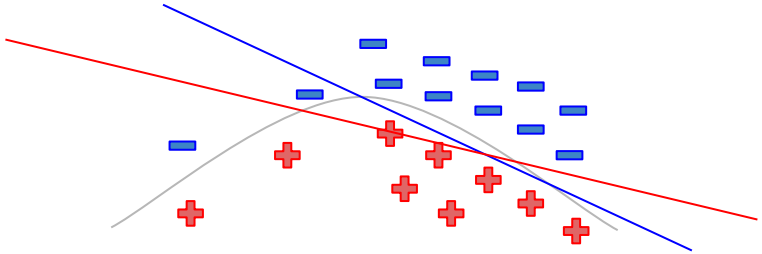
$$\hat{\epsilon}_S(w,x) = \frac{1}{N}\sum_{i=1}^{N} w(x_i) L(h_\theta(x_i^S), y_i^S)$$

$$\epsilon_T \leq \hat{\epsilon}_S(w,x) + \sqrt{\frac{O(\max_x w(x)^2)}{N}}$$

Source distribution:



Target distribution:



Proxy-labels methods

In previous case we use target domain data only for weight calculation. And do not use knowledge that target domain data contain.

It’s good to utilize somehow unlabelled data for training.

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{T} = \{x_i^T\}_{i=1}^M \sim P_T(x)$$

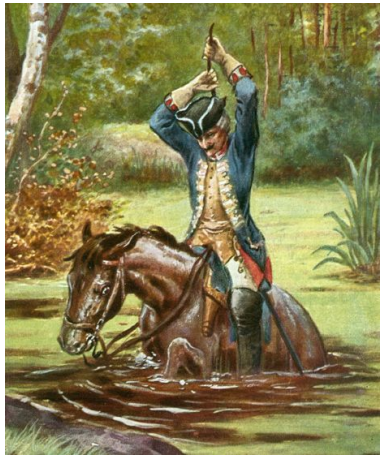
Self-training

Based on Ruder's presentation at ACL'18

1. Train model on labeled data.
2. Use confident predictions on unlabeled data as training examples. Repeat.

Algorithm 1 Self-training (Abney, 2007)

```
repeat
    m ← train_model(L)
    for x ∈ U do
        if max m(x) > τ then
            L ← L ∪ {(x, p(x))}
until no more predictions are confident
```

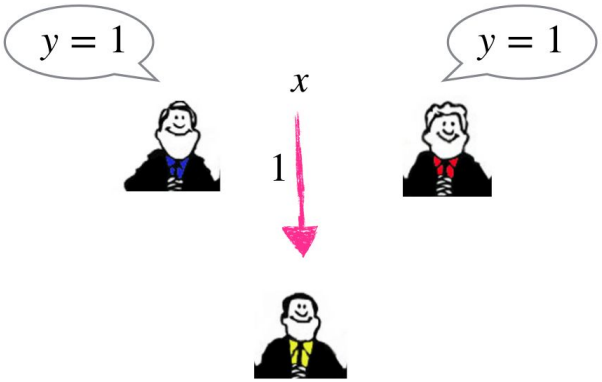


Error is amplified cause a model can not correct its own mistakes.

Tri-training

Based on Ruder's presentation at ACL'18

1. Train three models on bootstrapped samples.
2. Use predictions on unlabeled data for third if two agree.
3. Final prediction: majority voting



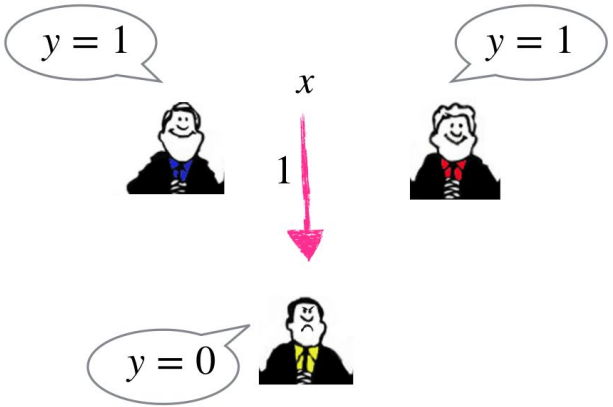
Algorithm 2 Tri-training (Zhou and Li, 2005)

```
for i ∈ {1..3} do
    S_i ← bootstrap_sample(L)
    m_i ← train_model(S_i)
repeat
    for i ∈ {1..3} do
        L_i ← ∅
        for x ∈ U do
            if p_j(x) = p_k(x)(j, k ≠ i) then
                L_i ← L_i ∪ {(x, p_j(x))}
        m_i ← train_model(L ∪ L_i)
until none of m_i changes
apply majority vote over m_i
```

Tri-training with disagreement

Based on [Ruder's presentation at ACL'18]

1. Train three models on bootstrapped samples.
2. Use predictions on unlabeled data for third if two agree and prediction differs.
3. Final prediction: majority voting

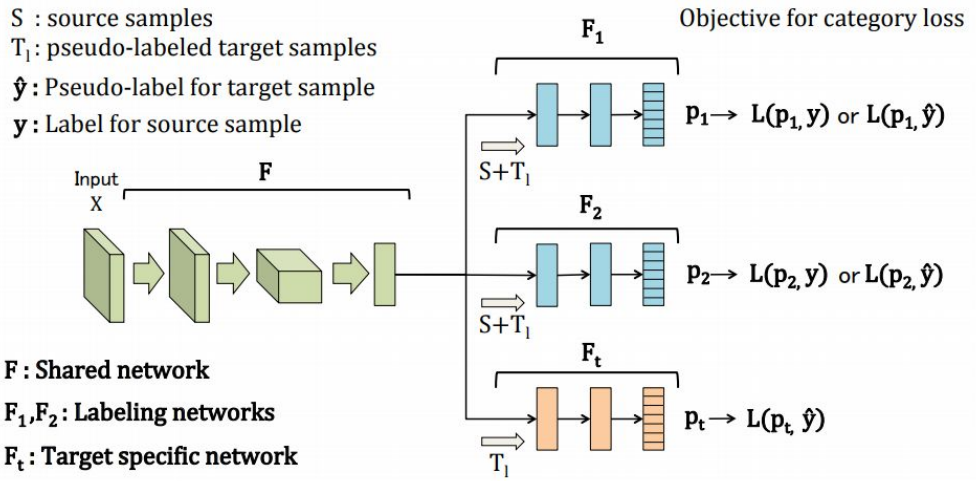


The problem with tri-training is that training three separate models can be too expensive.

Let's share parameters somehow.

Asymmetric Tri-training

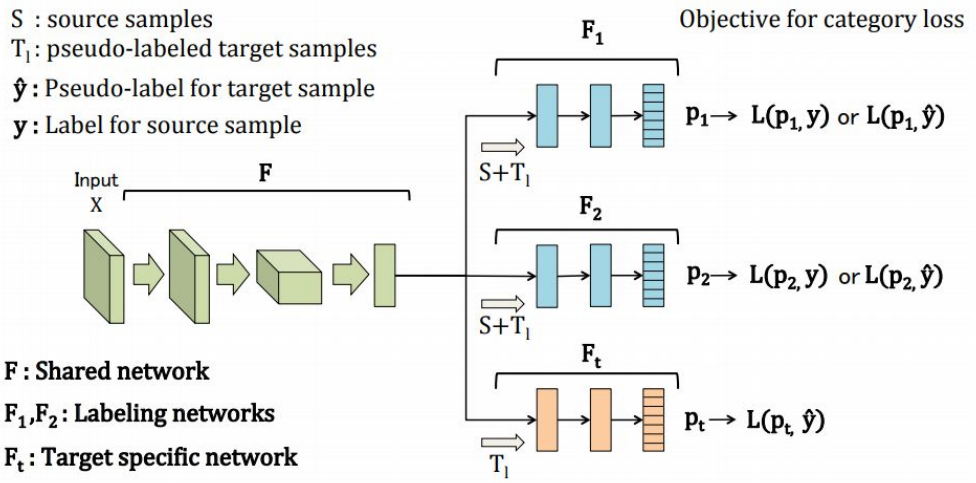
Saito et al., 2017



Input: data
$\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{m}, \mathcal{T} = \{(x_j)\}_{j=1}^{n}$
$\mathcal{T}_l = \emptyset$

Asymmetric Tri-training

Saito et al., 2017

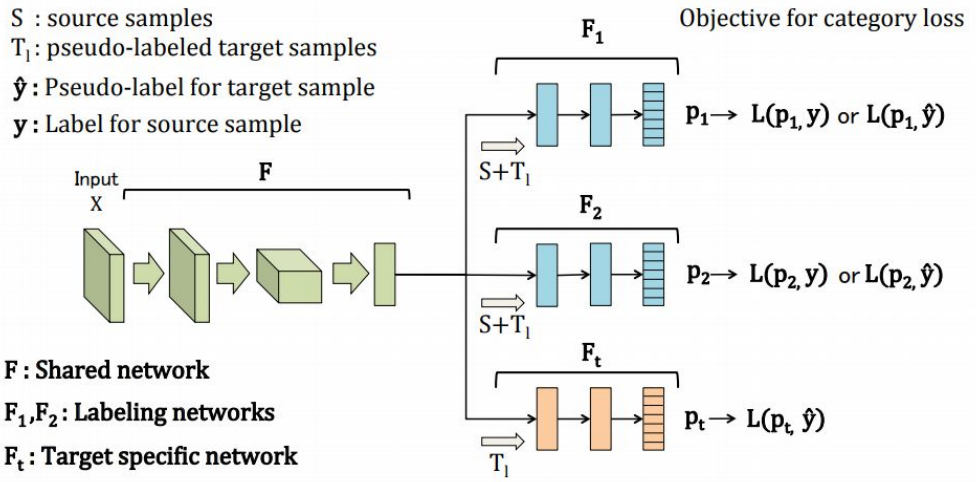


$$E(\theta_F, \theta_{F_1}, \theta_{F_2}) = \frac{1}{n}\sum_{i=1}^{n}\big[L_y(F_1 \circ F(x_i)), y_i) + L_y(F_2 \circ (F(x_i)), y_i)\big] + \lambda|{W_1}^T W_2|$$

Input: data
$\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{m}, \mathcal{T} = \{(x_j)\}_{j=1}^{n}$
$\mathcal{T}_l = \emptyset$
for $j = 1$ **to** $iter$ **do**
  Train F, F_1, F_2, F_t with mini-batch from training set $\mathcal{S}$
end for

Asymmetric Tri-training

Saito et al., 2017



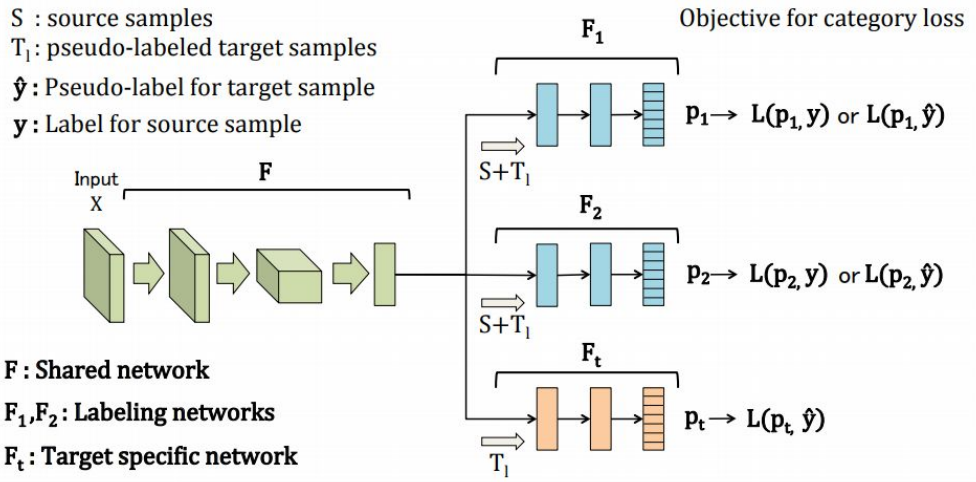
Input: data
$\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{m}, \mathcal{T} = \{(x_j)\}_{j=1}^{n}$
$\mathcal{T}_l = \emptyset$
for $j = 1$ **to** $iter$ **do**
  Train F, F_1, F_2, F_t with mini-batch from training set $\mathcal{S}$
end for

$$E(\theta_F, \theta_{F_1}, \theta_{F_2}) = \frac{1}{n}\sum_{i=1}^{n}\big[L_y(F_1 \circ F(x_i)), y_i) + L_y(F_2 \circ (F(x_i)), y_i)\big] + \boxed{\lambda |{W_1}^T W_2|}$$

To force F1 and F2 to learn from different features.

Asymmetric Tri-training

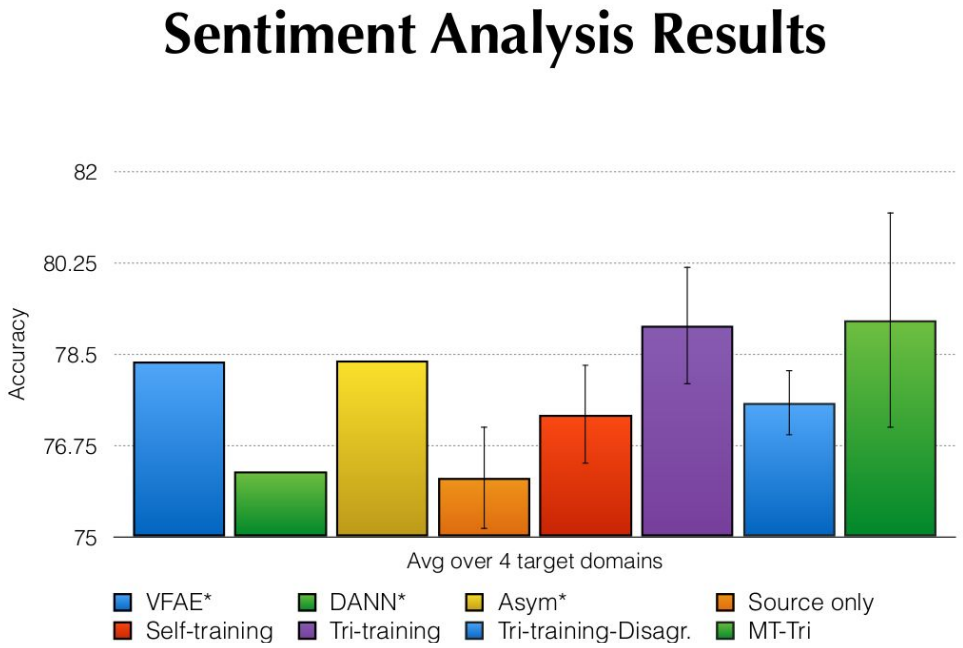
Saito et al., 2017



$$E(\theta_F, \theta_{F_1}, \theta_{F_2}) = \frac{1}{n}\sum_{i=1}^{n}\big[L_y(F_1 \circ F(x_i)), y_i) + L_y(F_2 \circ (F(x_i)), y_i)\big] + \lambda|{W_1}^T W_2|$$

Input: data
$\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{m}, \mathcal{T} = \{(x_j)\}_{j=1}^{n}$
$\mathcal{T}_l = \emptyset$
for $j = 1$ **to** $iter$ **do**
  Train F, F_1, F_2, F_t with mini-batch from training set $\mathcal{S}$
end for
$N_t = N_{init}$
$\mathcal{T}_l = \text{Labeling}(F, F_1, F_2, \mathcal{T}, N_t)$
$\mathcal{L} = \mathcal{S} \cup \mathcal{T}_l$
for k steps **do**
  for $j = 1$ **to** $iter$ **do**
    Train F, F_1, F_2 with mini-batch from training set $\mathcal{L}$
    Train F, F_t with mini-batch from training set $\mathcal{T}_l$
  end for
  $\mathcal{T}_l = \emptyset, N_t = k/20 * n$
  $\mathcal{T}_l = \text{Labeling}(F, F_1, F_2, \mathcal{T}, N_t)$
  $\mathcal{L} = \mathcal{S} \cup \mathcal{T}_l$
end for

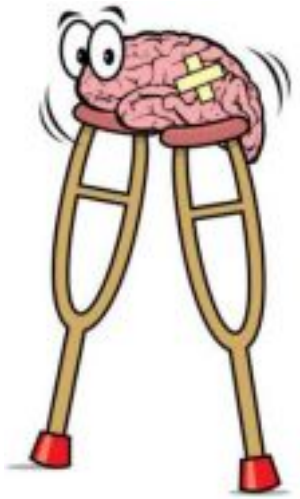
On effectiveness of proxy-label methods



Based on [Ruder's presentation at ACL'18]

Sentiment analysis on Amazon reviews
dataset (Blitzer et al, 2006)

Hack of the day...



Hack of the day: back-translation

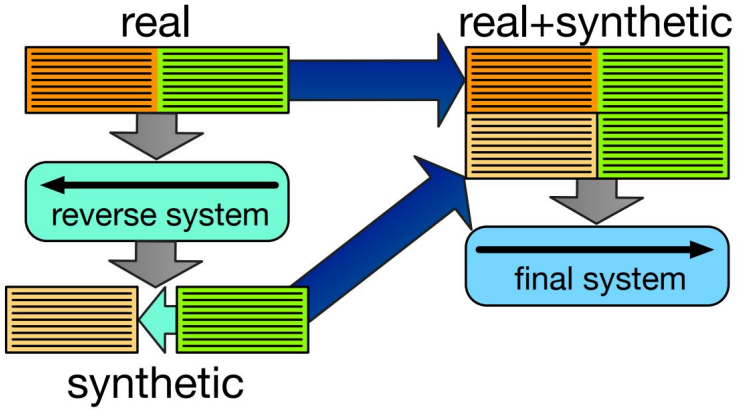
Sennrich, 2016b

In proxy-label methods we generate output labels based on unlabeled input data.

But what if we want to generate input data based on output? Does it make sense?

NMT is trained in the reverse translation direction (target-to-so then used to translate target-side monolingual data back into th language.

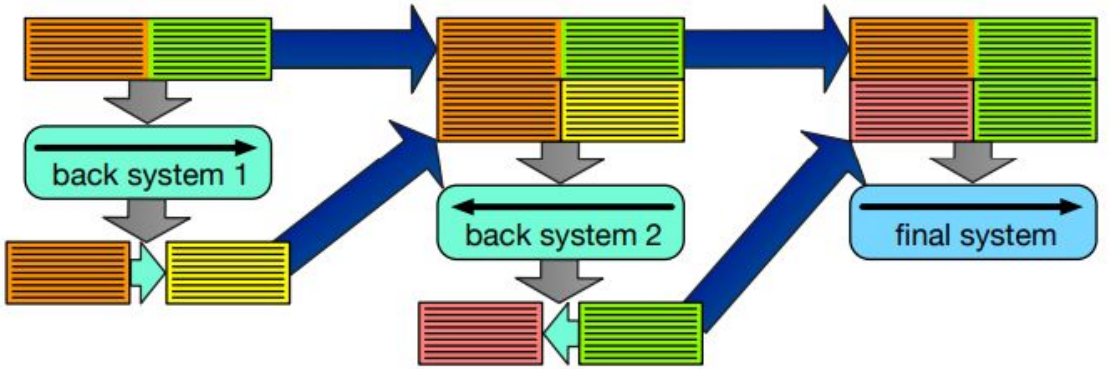
Picture from Hoang et al., 2018



Step further: iterative back-translation

Hoang et al., 2018

Iterative back-translation: back-translated data is used to build better translation systems in forward and backward directions, which in turn is used to reback-translate monolingual data



Wait a second…
Why do we need proxy-labels?
What about unsupervised pre-training?

Unsupervised in-domain pre-training is effective

Don't Stop Pretraining: Adapt Language Models to Domains and Tasks

Gururangan et al., 2020

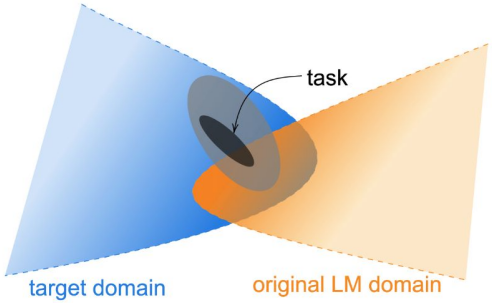


Figure 1: An illustration of data distributions. Task data is comprised of an observable task distribution, usually non-randomly sampled from a wider distribution (light grey ellipsis) within an even larger target domain, which is not necessarily one of the domains included in the original LM pretraining domain – though overlap is possible. We explore the benefits of continued pretraining on data from the task distribution and the domain distribution.

			Additional Pretraining Phases		
Domain	**Task**	**RoBERTa**	**DAPT**	**TAPT**	**DAPT + TAPT**
BioMed	ChemProt	$81.9_{1.0}$	$84.2_{0.2}$	$82.6_{0.4}$	$\mathbf{84.4}_{0.4}$
	†RCT	$87.2_{0.1}$	$87.6_{0.1}$	$87.7_{0.1}$	$\mathbf{87.8}_{0.1}$
CS	ACL-ARC	$63.0_{5.8}$	$75.4_{2.5}$	$67.4_{1.8}$	$\mathbf{75.6}_{3.8}$
	SciERC	$77.3_{1.9}$	$80.8_{1.5}$	$79.3_{1.5}$	$\mathbf{81.3}_{1.8}$
News	HyperPartisan	$86.6_{0.9}$	$88.2_{5.9}$	$\mathbf{90.4}_{5.2}$	$90.0_{6.6}$
	†AGNews	$93.9_{0.2}$	$93.9_{0.2}$	$94.5_{0.1}$	$\mathbf{94.6}_{0.1}$
Reviews	†Helpfulness	$65.1_{3.4}$	$66.5_{1.4}$	$68.5_{1.9}$	$\mathbf{68.7}_{1.8}$
	†IMDB	$95.0_{0.2}$	$95.4_{0.1}$	$95.5_{0.1}$	$\mathbf{95.6}_{0.1}$

Table 5: Results on different phases of adaptive pretraining compared to the baseline RoBERTa (col. 1). Our approaches are DAPT (col. 2, §3), TAPT (col. 3, §4), and a combination of both (col. 4). Reported results follow the same format as Table 3. State-of-the-art results we can compare to: ChemProt (84.6), RCT (92.9), ACL-ARC (71.0), SciERC (81.8), HyperPartisan (94.8), AGNews (95.5), IMDB (96.2); references in §A.2.

Unsupervised in-domain pre-training is effective

Don't Stop Pretraining: Adapt Language Models to Domains and Tasks

Gururangan et al., 2020

Pretraining	BIOMED		CS
	CHEMPROT	RCT-500	ACL-ARC
ROBERTA	$81.9_{1.0}$	$79.3_{0.6}$	$63.0_{5.8}$
TAPT	$82.6_{0.4}$	$79.8_{1.4}$	$67.4_{1.8}$
RAND-TAPT	$81.9_{0.6}$	$80.6_{0.4}$	$69.7_{3.4}$
50NN-TAPT	$83.3_{0.7}$	$80.8_{0.6}$	$70.7_{2.8}$
150NN-TAPT	$83.2_{0.6}$	$81.2_{0.8}$	$73.3_{2.7}$
500NN-TAPT	$83.3_{0.7}$	$81.7_{0.4}$	$\mathbf{75.5}_{1.9}$
DAPT	$\mathbf{84.2}_{0.2}$	$\mathbf{82.5}_{0.5}$	$75.4_{2.5}$

Table 8: Mean test set micro-F_1 (for CHEMPROT and RCT) and macro-F_1 (for ACL-ARC), across five random seeds, with standard deviations as subscripts, comparing RAND-TAPT (with 50 candidates) and kNN-TAPT selection. Neighbors of the task data are selected from the domain data.

Unsupervised in-domain pre-training is effective

Don't Stop Pretraining: Adapt Language Models to Domains and Tasks

Han et al., 2020

AdaptaBERT:

1. Domain tuning: (50/50 source+target data unsupervised pretraining.
2. Task tuning using source data

	WNUT				CoNLL
System	domain adaptation data	Precision	Recall	**F1**	F1
Unsupervised domain adaptation					
1. Task-tuned BERT	n/a	50.9	66.6	57.7	**97.8**
2. AdaptaBERT	WNUT training	52.8	66.7	58.9	97.6
3. AdaptaBERT	+ 1M tweets	53.6	68.3	60.0	97.8
4. AdaptaBERT	WNUT train+test	57.7	68.9	**62.8**	97.8
Supervised in-domain training					
5. Fine-tuned BERT	n/a	66.3	62.3	64.3	80.9
6. Limsopatham and Collier (2016)	n/a	73.5	59.7	**65.9**	

Table 4: Named entity segmentation performance on the WNUT test set and CoNLL test set A. Limsopatham and Collier (2016) had the winning system at the 2016 WNUT shared task. Their results are reprinted from their paper, which did not report performance on the CoNLL dataset.

(Semi)-supervised domain adaptation

Both in source and target domain labels are known.

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Fine-tuning

- Train model on the target domain data
- Train with lower learning rate on the source domain

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Fine-tuning

- Train model on the target domain data
- Train with lower learning rate on the source domain

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Problems?

Fine-tuning

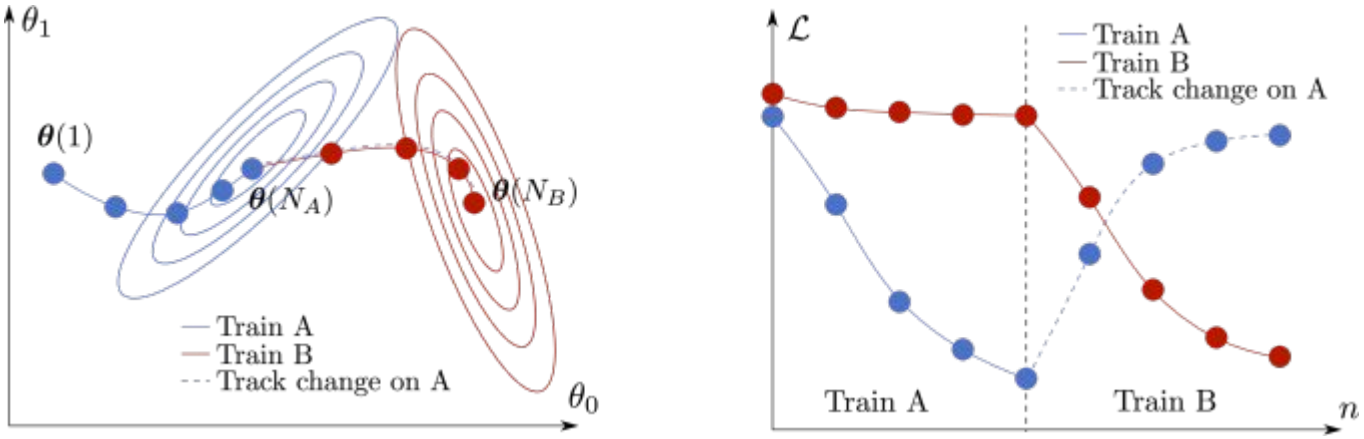
- Train model on the labeled target domain data
- Train with lower learning rate on the labeled source domain data

$$\mathcal{D}_S = \{x_i^S, y_i^S\}_{i=1}^N \sim P_S(x, y)$$

$$\mathcal{D}_T = \{x_i^T, y_i^T\}_{i=1}^M \sim P_T(x, y)$$

Ordinarily, in-domain data is limited, therefore we have generalization problems (over-fitting).

Fine-tuning: catastrophic forgetting



Picture from Felix Wiewel, Bin Yang, 2019

Fine-tuning: catastrophic forgetting

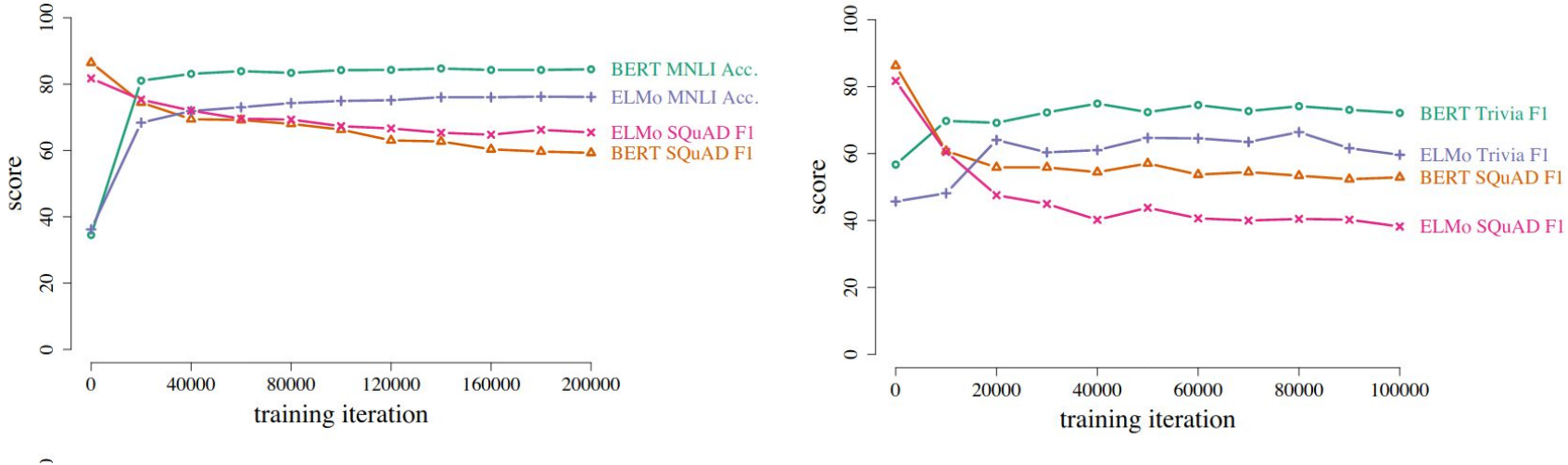


Figure 4: Catastrophic forgetting in a continual learning setup on unsupervised → SQuAD → MNLI (top) and unsupervised → SQuAD → TriviaQA (bottom).

From *Learning and Evaluating General Linguistic Intelligence* [Yogotama et al., 2019]

Fine-tuning: regularization techniques

Ordinarily, in-domain data is limited, therefore we have generalization problems (over-fitting). To prevent this:

Fine-tuning: regularization techniques

Ordinarily, in-domain data is limited, therefore we have generalization problems (over-fitting). To prevent this:

1. Weight decay $L_W = ||W||_{L_p}$

W is the in-domain parameter matrix to be learned

$\hat{W}$ is the corresponding fixed out-of-domain parameter matrix.

Fine-tuning: regularization techniques

Ordinarily, in-domain data is limited, therefore we have generalization problems (over-fitting). To prevent this:

1. Weight decay $L_W = ||W||_{L_p}$
2. Dropout

W is the in-domain parameter matrix to be learned

$\hat{W}$ is the corresponding fixed out-of-domain parameter matrix.

Fine-tuning: regularization techniques

Ordinarily, in-domain data is limited, therefore we have generalization problems (over-fitting). To prevent this:

1. Weight decay $L_W = ||W||_{L_p}$
2. Dropout
3. L2-distance from out-of-domain penalization (MAP-L2) $L_W = \lambda \cdot \left\| W - \hat{W} \right\|_2^2$

W is the in-domain parameter matrix to be learned

$\hat{W}$ is the corresponding fixed out-of-domain parameter matrix.

Fine-tuning: regularization techniques

Barone et al., 2017

Out-of-domain: WMT
In-domain: TED talks

Table 1: English-to-German translation BLEU scores

	valid	test			
System	tst2010	tst2011	tst2012	tst2013	avg
Out-of-domain only	27.19	29.65	25.78	27.85	27.76
In-domain only	25.95	27.84	23.68	25.83	25.78
Fine-tuning	30.53	32.62	28.86	32.11	31.20
Fine-tuning + dropout	30.63	33.06	28.90	32.02	31.33
Fine-tuning + MAP-L2	30.81	32.87	28.99	31.88	31.25
Fine-tuning + tuneout	30.49	32.07	28.66	31.60	30.78†
Fine-tuning + dropout + MAP-L2	30.80	**33.19**	**29.13**	**32.13**	**31.48†**

†: different from the fine-tuning baseline at 5% significance.

Limited degradation on out-of-domain

L2-distance from out-of-domain penalization (MAP-L2) limits quality degradation on the out-of-domain.

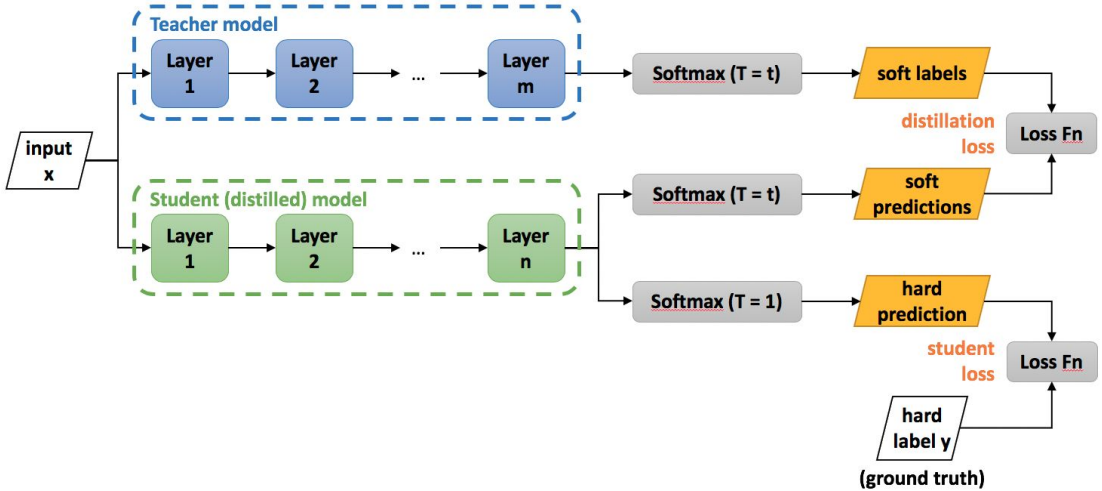
Limited degradation on out-of-domain

L2-distance from out-of-domain penalization (MAP-L2) limits quality degradation on the out-of-domain.

But we can directly force predictions of in-domain model and out-of-domain model to be closer to each other!

Distillation-like domain adaptation

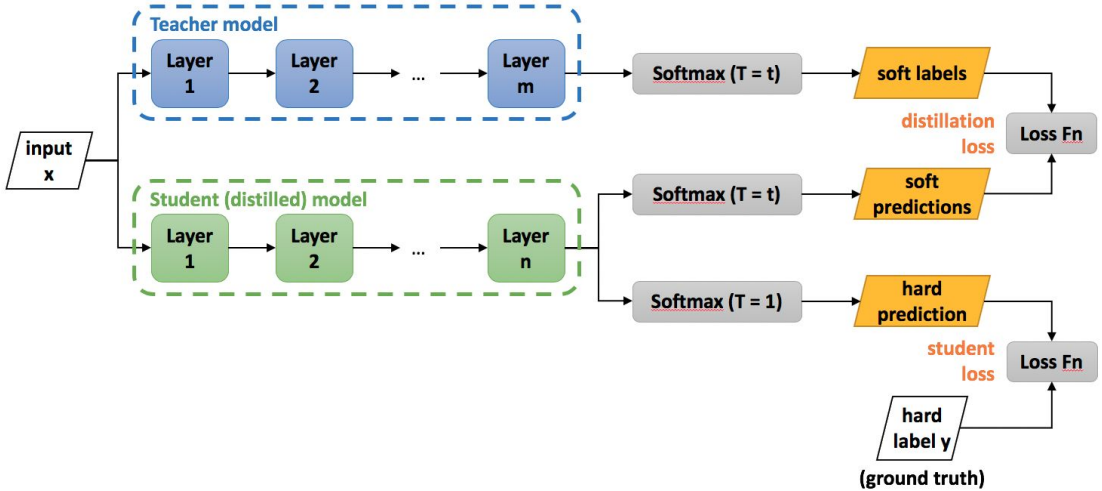
“Knowledge distillation” framework of (Hinton et al., 2014): a smaller “student” network learns to mimic a large “teacher” network by minimizing the loss between the output distributions of the two networks.



Picture from nervanasystems.github.io

Distillation-like domain adaptation

“Knowledge distillation” framework of (Hinton et al., 2014): a smaller “student” network learns to mimic a large “teacher” network by minimizing the loss between the output distributions of the two networks.



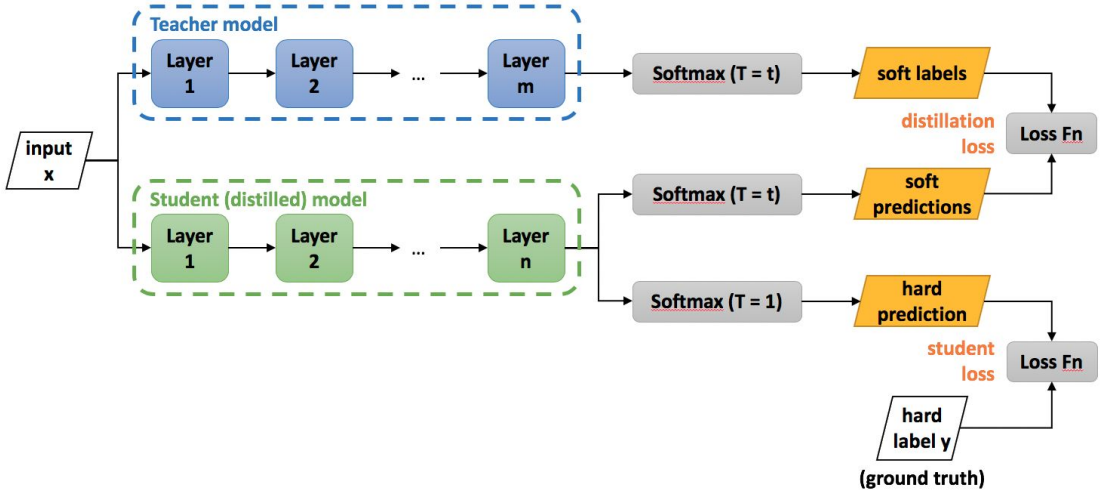
Originally, to compress multiple models (ensemble) to a smaller model.

Picture from nervanasystems.github.io

Distillation-like domain adaptation

Hinton et al., 2015

"Knowledge distillation" framework of (Hinton et al., 2014): a smaller "student" network learns to mimic a large "teacher" network by minimizing the loss between the output distributions of the two networks.

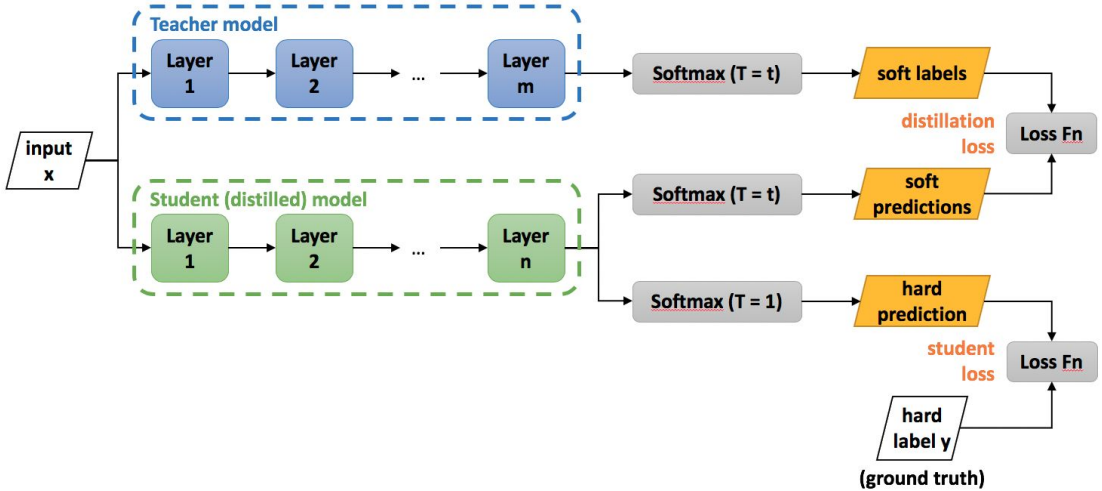


But this framework can be adapted for domain adaptation.

Picture from nervanasystems.github.io

Distillation-like domain adaptation

- Train teacher network
- Initialize student network by weights of teacher
- Train student with composite loss



[Dakwale et al., 2017]

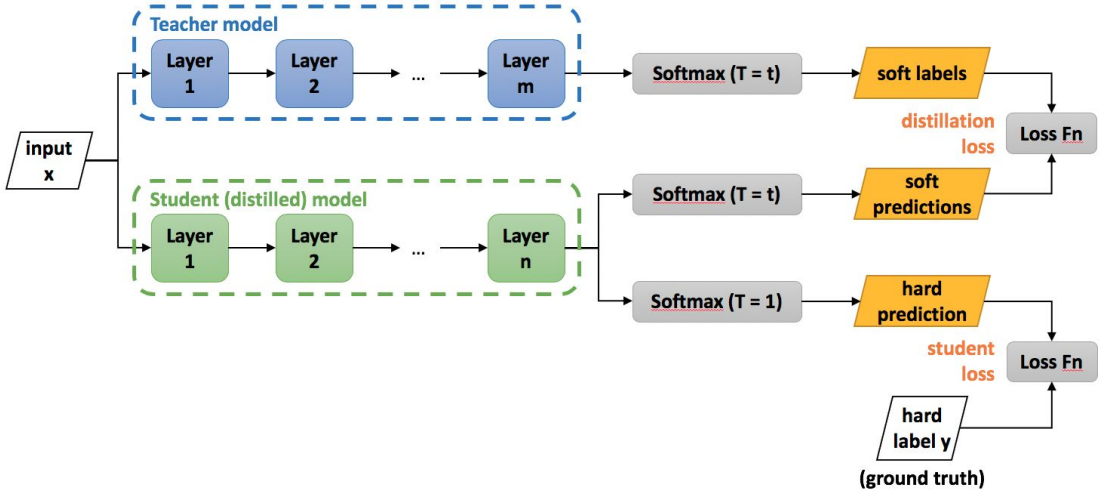
Teacher produces distribution *p*,
Student produces distribution *q*.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Picture from nervanasystems.github.io

Distillation-like domain adaptation

- Train teacher network
- Initialize student network by weights of teacher
- Train student with composite loss



Composite loss consists of two terms:

1. Cross-entropy loss on *q*
2. KL-divergence between *p* and *q*

$$L_{total} = (1-\lambda)L(q_\theta) + \lambda D_{KL}(p||q_\theta)$$

Picture from nervanasystems.github.io

Distillation-like domain adaptation

Composite loss consists of two terms:

1. Cross-entropy loss between hard-label distribution and *q*
2. KL-divergence between *p* and *q*

$$L_{total} = (1 - \lambda)L(q_\theta) + \lambda D_{KL}(p||q_\theta)$$

Distillation-like domain adaptation

Composite loss consists of two terms:

1. Cross-entropy loss between hard-label distribution and *q*
2. KL-divergence between *p* and *q*

$$L_{total} = (1 - \lambda)L(q_\theta) + \lambda D_{KL}(p||q_\theta)$$

$$L(q_\theta) = -\sum_k [l = y_k] \log q_\theta(y_k) = H(l, q_\theta)$$

Distillation-like domain adaptation

Composite loss consists of two terms:

1. Cross-entropy loss between hard-label distribution and *q*
2. KL-divergence between *p* and *q*

$$L_{total} = (1 - \lambda)L(q_\theta) + \lambda D_{KL}(p||q_\theta)$$

$$D_{KL}(p||q_\theta) = \sum_k p(y_k)(\log p(y_k) - \log q_\theta(y_k)) = H(p, q_\theta) - H(p)$$

Distillation-like domain adaptation

Composite loss consists of two terms:

1. Cross-entropy loss between hard-label distribution and *q*
2. KL-divergence between *p* and *q*

$$L_{total} = (1-\lambda)L(q_\theta) + \lambda D_{KL}(p||q_\theta)$$

$$L_{total} \sim (1-\lambda)H(l, q_\theta) + \lambda(H(p, q_\theta) - H(p)) \sim (1-\lambda)H(l, q_\theta) + \lambda H(p, q_\theta)$$

Distillation-like domain adaptation

Using composite loss we force internal representation to be tolerant to domain. Thus we implicitly minimize discrepancy between domain distributions in internal representation space.

$$\epsilon_T \leq \epsilon_S + \boxed{\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)} + \lambda$$

The divergence between source and target domain

Should we minimize distribution divergence directly?

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

The divergence between source and target domain

Should we minimize distribution divergence directly?

Deep distribution alignment! -> adversarial methods, variational bayes methods and other cool stuff.

$$\epsilon_T \leq \epsilon_S + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

The divergence between source and target domain

Batch normalization revisited

Li et al., 2017

Algorithm 1 Adaptive Batch Normalization (AdaBN)

for neuron j in DNN **do**

Concatenate neuron responses on all images of target domain t: $\mathbf{x}_j = [\ldots, x_j(m), \ldots]$

Compute the mean and variance of the target domain: $\mu_j^t = \mathbb{E}(\mathbf{x}_j^t)$, $\sigma_j^t = \sqrt{\mathrm{Var}(\mathbf{x}_j^t)}$.

end for

for neuron j in DNN, testing image m in target domain **do**

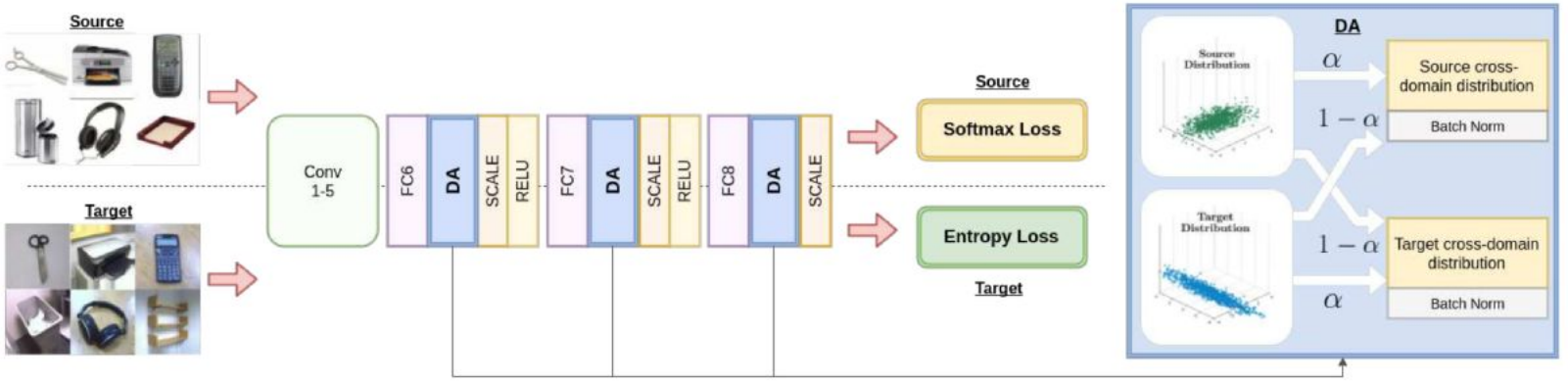
Compute BN output $y_j(m) := \gamma_j \frac{\left(x_j(m) - \mu_j^t\right)}{\sigma_j^t} + \beta_j$

end for

$$\mathtt{DA}(x_s;\alpha) = \frac{x_s - \mu_{st,\alpha}}{\sqrt{\epsilon + \sigma_{st,\alpha}^2}}, \quad \mathtt{DA}(x_t;\alpha) = \frac{x_t - \mu_{ts,\alpha}}{\sqrt{\epsilon + \sigma_{ts,\alpha}^2}},$$

Batch normalization revisited

Carlucci et al., 2017



$$\mathtt{DA}(x_s;\alpha) = \frac{x_s - \mu_{st,\alpha}}{\sqrt{\epsilon + \sigma^2_{st,\alpha}}}, \quad \mathtt{DA}(x_t;\alpha) = \frac{x_t - \mu_{ts,\alpha}}{\sqrt{\epsilon + \sigma^2_{ts,\alpha}}},$$

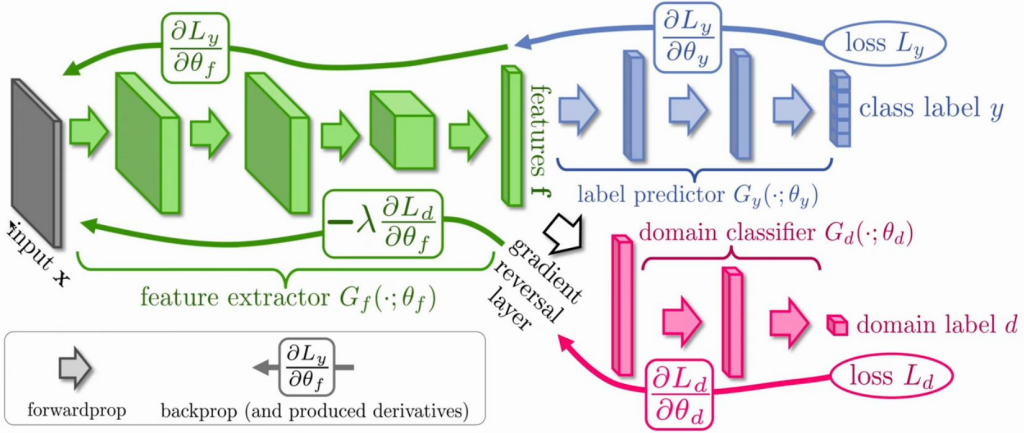
Assume q^s and q^t to be the distribution of x_s and x_t, respectively, and let $q^{st}_\alpha = \alpha q^s + (1-\alpha)q^t$ and, symmetrically, $q^{ts}_\alpha = \alpha q^t + (1-\alpha)q^s$ be cross-domain distributions mixed by a factor $\alpha \in [0.5, 1]$.

Auto-DIAL

(slide from Alex Bronstein lecture)

Deep distribution alignment

- Train a classifier for the **domain** of an image based on deep convolutonal features.
- Try to maximize the loss of this classifier when training the CNN (**confusion loss**).
- Simultaneously, minimize the classification loss on the source domain using the same convolutional features.
- Train the digit classifier with source domain data, and the domain classifier with both domains' data.



Ganin, Lempitsky 2015

Deep distribution alignment

Long et al., 2015

Domain adaptation network utilizes Maximum Mean Discrepancy (MMD)

$$d_k^2(p,q) \triangleq \left\| \mathbf{E}_p\left[\phi(\mathbf{x}^s)\right] - \mathbf{E}_q\left[\phi(\mathbf{x}^t)\right] \right\|_{\mathcal{H}_k}^2$$

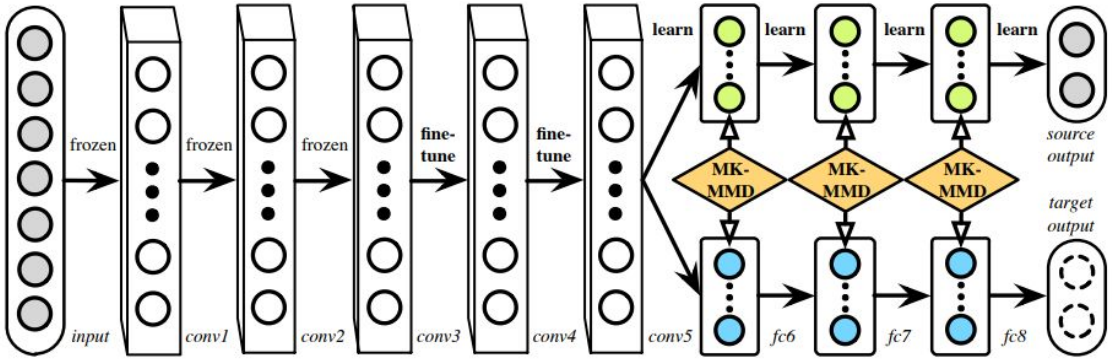


Figure 1. The DAN architecture for learning transferable features.