Domain Adaptation

CS 330

Course Reminders

Optional homework 4 due next **Monday**.

Project milestone due next **Wednesday**

Azure: If you are close to running out of credits, proactively request more in private Ed post.

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Problem Settings Recap

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task(s) $\mathcal{T}_a$ by *transferring* knowledge learned from $\mathcal{T}_a$

Meta-Learning Problem

Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

What is domain adaptation?

→ Similar to MTL & Transfer Learning, but somewhat of a special case.

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$ → Possibly multiple source domains

A form of transfer learning, with access to target domain data during training ✱

("transductive" learning)

→ Key difference!

⇒ This is usually the case, except that I didn't use the target domain data simultaneously (for transfer learning)

→ Scenarios where domain adaptation is helpful
→ little to no data in the target domain

Unsupervised domain adaptation: access to unlabeled target domain data

Semi-supervised domain adaptation: access to unlabeled and labeled target domain data

Supervised domain adaptation: access to labeled target domain data. → Could do pretty well using just fine-tuning (transfer learning)

We will focus on ***unsupervised domain adaptation***.

What is domain adaptation?

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer learning, with **access to target domain data during training** ("transductive" learning)

Unsupervised domain adaptation: access to unlabeled target domain data

Common assumptions:

- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error.

A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

- X는 사람, Y는 그 사람의 국적, domain을 국가라고 하면, $p_{\text{한국}}(\mathbf{x})$, $p_{\text{미국}}(\mathbf{x})$를 각 나라에 대한 사람의 분포라고 볼 수 있음
  - Domain shift 예시 (조금 많이 shift 된 것 같다...)

- $x \sim p_{\text{한국}}(\mathbf{x})$
  (한국에서 샘플링한 한국인)

- $x \sim p_{\text{미국}}(\mathbf{x})$
  (미국에서 샘플링한 한국인)

- 각각을 task로 바라볼 경우 : $p(y|x)$가 domain에 따라 변하게 됨
  (task마다 고려하는 한국인의 기준이 다르니까)

$$p_{\text{한국}}(y = \text{korean}|x) = 0.9$$
$$p_{\text{미국}}(y = \text{korean}|x) = 0.6$$

$$p_{\text{한국}}(y = \text{korean}|x) = 0.4$$
$$p_{\text{미국}}(y = \text{korean}|x) = 0.8$$

- 각각을 domain으로 바라볼 경우 : $p(y|x)$가 domain이 변하더라도 바뀌지 않아야 함
  (domain만 바뀌었을 뿐, 한국인이라는 것 자체는 변함이 없다고 보는 것)

$$p_{\text{한국}}(y = \text{korean}|x) = p_{\text{미국}}(y = \text{korean}|x) = 0.8$$

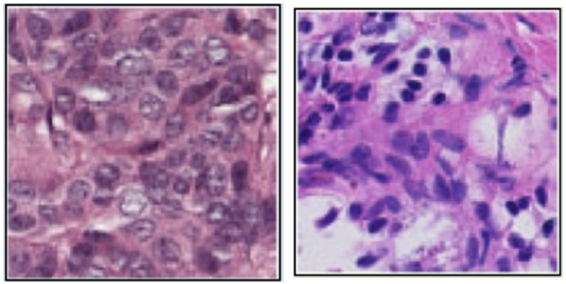
$$p_{\text{한국}}(y = \text{korean}|x) = p_{\text{미국}}(y = \text{korean}|x) = 0.8$$

Example domain adaptation problems

Satellite image → land use prediction paper
→ also domain adaptation? → Yes

Tumor detection & classification

Source hospital Target hospital



varying imaging techniques, different demographics

Essentially predicting the same target var.
→ only P(x) has changed (P(y|x) const.)

i.e. different ways of collecting the data

Land use classification

Source region Target region

appearance of buildings, plants; weather conditions, pollution

Text classification, generation

Source corpus Target corpus

Simple English WIKIPEDIA

arXiv

PubMed

differing sentence structure, vocabulary, word use

But this target remains the same.

Domains can also be:

- people/users
- points in time
- institutions (schools, companies, universities)

Revisiting assumptions:

- Access to target domain data during training.
- There exists a single hypothesis $f(y \mid x)$ with low error.

(*) This is an important distinction.

1. 병원에서 악성 종양을 판단하는 AI를 훈련하려 한다.
2. 이를 위해 A라는 병원은 이미지에서 종양을 감지할 수 있도록 Classifier 훈련시켰다. ***(Source domain)***
3. 이 모델을 B라는 병원에 배포하여 같은 기능을 수행하게 하고 싶다. ***(Target domain)***
4. 하지만 B 병원은 A병원과 비교할 때 이미지를 수집하는 기술이 다르거나 환자의 인구 통계가 다를 수 있다. 따라서 A 병원이 추출한 종양 이미지와 B 병원이 추출한 종양 이미지는 조금 다를 수 있다. ***(Distribution shift)***
5. Target domain의 트레이닝 전 Classifier와는 달리 B 병원의 의사는 여전히 환자의 종양 이미지를 보고 종양이 있는지 여부를 판단할 수 있고 이는 판단에 필요한 단일 기능이 존재한다는 것을 알 수 있다. $p(y|x), \mathcal{L}$
6. 이를 두고 A 병원(Task 1) 과 B병원(Task 2)이 서로 다른 task를 수행한다고 볼 수도 있다. 다만 $p_i(x)$ 만이 다른 경우이다.

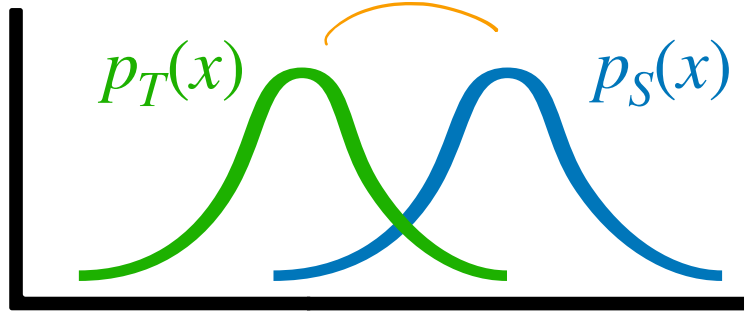
Plan for Today

Domain Adaptation

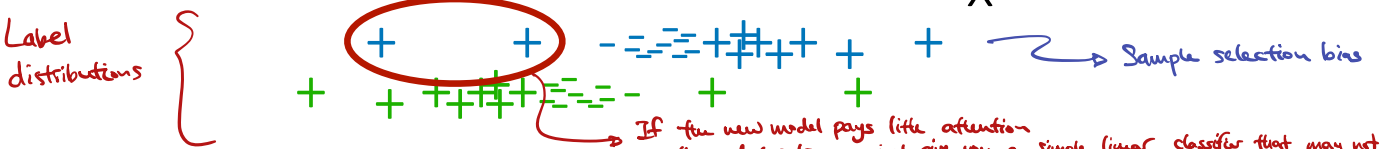
- Problem statements
- Algorithms
  - **Data reweighting**
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Toy domain adaptation problem



e.g. sample selection bias



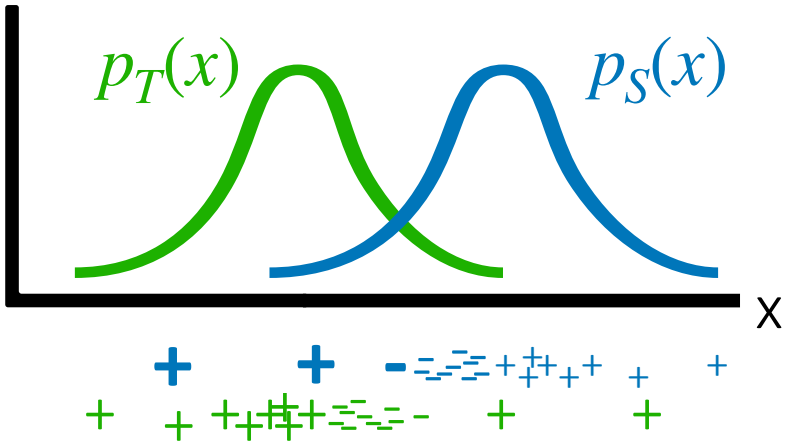
Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

How can we learn a classifier that does well on $p_T(x)$?

(using labeled data from $p_S(x)$ & unlabeled data from $p_T(x)$)

Problem adapted from Blitzer & Daume ICML '10

Toy domain adaptation problem



e.g. sample selection bias

Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

Solution: Upweight examples with high $p_T(x)$ but low $p_S(x)$

Why does this make sense mathematically?

Problem adapted from Blitzer & Daume ICML '10

$$\min_\theta \mathbb{E}_{P_T(x,y)}\left[\mathcal{L}(f_\theta(x), y)\right] = \int P_T(x,y)\,\mathcal{L}(f(x), y)\,\frac{P_S(x,y)}{P_S(x,y)} = \mathbb{E}_{P_S(x,y)}\left[\frac{P_T(x,y)}{P_S(x,y)}\,\mathcal{L}(f(x), y)\right]$$

- $\frac{P_S(x,y)}{P_S(x,y)}$: Just 1.
- $\mathbb{E}_{P_T(x,y)}$: How well we do on the target dist.
  - We can't sample from this dist. directly, but we can sample from $P_S(x)$ (source dist)
- $\mathbb{E}_{P_S(x,y)}$: How well we do on the source dist.
- $\frac{P_T(x,y)}{P_S(x,y)}$: weighting our loss based on this ratio ⊛
  - ⊛ : (High likelihood on the target) or (Low likelihood on the source) → Upweighted
  - (Low likelihood on the target) or (High likelihood on the source) → downweighted

ERM: Empirical Risk Minimization

$\min \mathbb{E}_{P_S(x)}[\mathcal{L}]$

Using the assumption: $P_T(y|x) = P_S(y|x)$ → There are some problems where this will hold, & there are others where it will not.

$P(x,y) = P(x)\cdot P(y|x)$

$\Rightarrow P_T(x,y) = P_T(x)\cdot P_T(y|x)$, $P_S(x,y) = P_S(x)\cdot P_S(y|x)$ → Identical, cancels out

$$= \mathbb{E}_{P_S(x,y)}\left[\frac{P_T(x)}{P_S(x)}\,\mathcal{L}(f(x), y)\right]$$

no y's here!

Q: How do we estimate this ratio $\frac{P_T(x)}{P_S(x)}$?

$$P_T(x) = P(x|d=tgt) = \frac{P(d=tgt|x)\cdot P(x)}{P(d=tgt)}$$ (& likewise for source domain)

$$\frac{P_T(x)}{P_S(x)} = \frac{P(x|d=tgt)}{P(x|d=src)} = \frac{P(d=tgt|x)\cdot P(x)}{P(d=tgt)}\cdot\frac{P(d=src)}{P(d=src|x)\cdot P(x)}$$

$$= \frac{P(d=tgt|x)}{P(d=src|x)}\cdot\frac{P(x)}{P(x)}\cdot\frac{P(d=src)}{P(d=tgt)}$$

- $\frac{P(d=tgt|x)}{P(d=src|x)}$: Can estimate using discriminative model
  - Train a classifier to predict if an example came from the target or the source domain : $x \to \hat{y}$, $tgt|src$, $P(tgt|x) = 1 - P(src|x)$
- $\frac{P(x)}{P(x)} = 1$
- $\frac{P(d=src)}{P(d=tgt)}$: Const. (doesn't depend on θ)

Domain adaptation via importance sampling

Empirical risk minimization on source data: $\min_\theta \mathbb{E}_{p_S(x,y)}[L(f_\theta(x), y)]$

Goal: ERM on target distribution: $\min_\theta \mathbb{E}_{p_T(x,y)}[L(f_\theta(x), y)]$

$$\mathbb{E}_{p_T(x,y)}[L(f_\theta(x), y)] = \int p_T(x, y) L(f_\theta(x), y) dx dy$$

$$= \int p_T(x, y) \frac{p_S(x, y)}{p_S(x, y)} L(f_\theta(x), y) dx dy$$

$$= \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x, y)}{p_S(x, y)} L(f_\theta(x), y) \right]$$

Note: $p(y \mid x)$ cancels out if it is the same for source & target

Solution: Upweight examples with high $p_T(x)$ but low $p_S(x)$

Domain adaptation via importance sampling

$$\min_\theta \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y)\right]$$

How to estimate the importance weights $\frac{p_T(x)}{p_S(x)}$?

Option 1: Estimate likelihoods $p_T(x)$ and $p_S(x)$, then divide. But, difficult to estimate accurately.

Can we estimate the ratio ***without*** training a generative model?

Bayes rule:

$$p(x \mid \text{target}) = \frac{p(\text{target} \mid x)p(x)}{p(\text{target})}$$

$$p(x \mid \text{source}) = \frac{p(\text{source} \mid x)p(x)}{p(\text{source})}$$

$$\frac{p_T(x)}{p_S(x)} = \frac{p(x \mid \text{target})}{p(x \mid \text{source})} = \frac{p(\text{target} \mid x)p(\text{source})}{p(\text{source} \mid x)p(\text{target})}$$

↑ can estimate with binary classifier!

↑ a constant

Bickel, Bruckner, Scheffer. Discriminative Learning Under Covariate Shift. JMLR '09

Domain adaptation via importance sampling

$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

$$\frac{p_T(x)}{p_S(x)} = \frac{p(x \mid \text{target})}{p(x \mid \text{source})} = \frac{p(\text{target} \mid x) p(\text{source})}{p(\text{source} \mid x) p(\text{target})}$$

can estimate with binary classifier!

a constant

Full algorithm:

1. Train binary classifier $c(\text{source} \mid x)$ to discriminate between source and target data.
2. Reweight or resample data $\mathcal{D}_S$ according to $\frac{1 - c(\text{source} \mid x)}{c(\text{source} \mid x)}$.
3. Optimize loss $L(f_\theta(x), y)$ on reweighted or resampled data.

Bickel, Bruckner, Scheffer. Discriminative Learning Under Covariate Shift. JMLR '09

What assumption does this make?

$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

Source $p_S(x)$ needs to cover the target $p_T(x)$.

Formally: if $p_T(x) \neq 0$, then $p_S(x) \neq 0$.

- Important assumption.
  - ⇒ If this is satisfied, then there are theoretical guarantees for this method.
- But this may not hold, depending on domains.

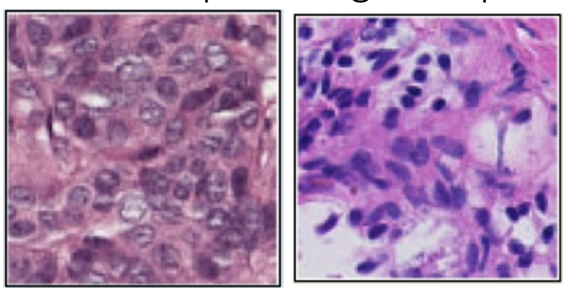
Text classification, generation

Source corpus Target corpus

—> May have enough coverage of distr.

Tumor detection & classification

Source hospital Target hospital



—> Source probably won't cover target distr!

↳ i.e. images look too different for there to be an overlap ($P_S(x)$ covering $P_T(x)$).

→ Another approach is needed.

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - **Feature alignment**
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

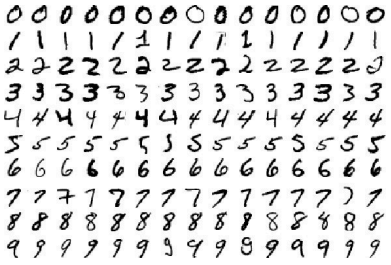
Domain adaptation if support is not shared?

region of the dist. for which density is nonzero.

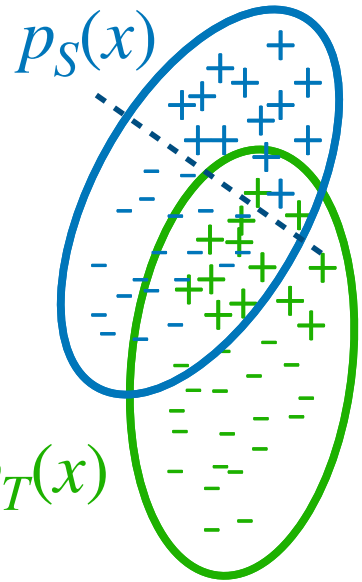
If $p_S(x)$ does not cover $p_T(x)$.

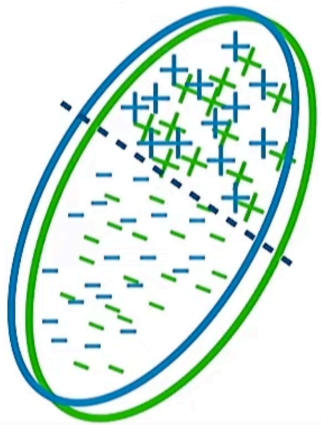
encourage it to have similar representations of the two dist.

Can we align the features?



Images taken from street views.





Source classifier in *aligned feature space* is more accurate in target domain.

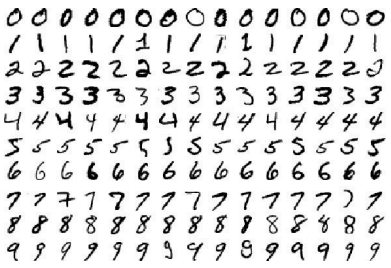
How to align the features?

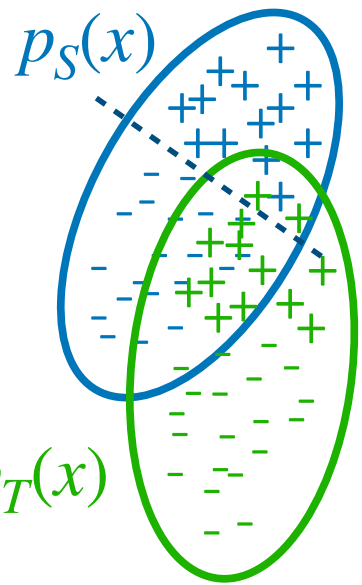
Having a more interesting/characteristic manifold in the source & target data distributions increases the chances this algo works successfully.

eg)

Finding rotation w/o the target domain labels is difficult.

Domain adaptation if support is not shared?





How to align the features?

Source encoder f_{θ_S} Target encoder f_{θ_T}

or share a part of params btw the two

↳ Could use the same encoder for both source & target

Need to match features at ***population-level***.

↳ as opposed to looking at matching individual examples

i.e. make encoded samples $f_{\theta_S}(x), x \sim p_S(\cdot)$

indistinguishable from $f_{\theta_T}(x), x \sim p_T(\cdot)$

Key idea: Try to fool a domain classifier $c(d = \text{source} \,|\, f(x))$.

↳ features extracted from x.

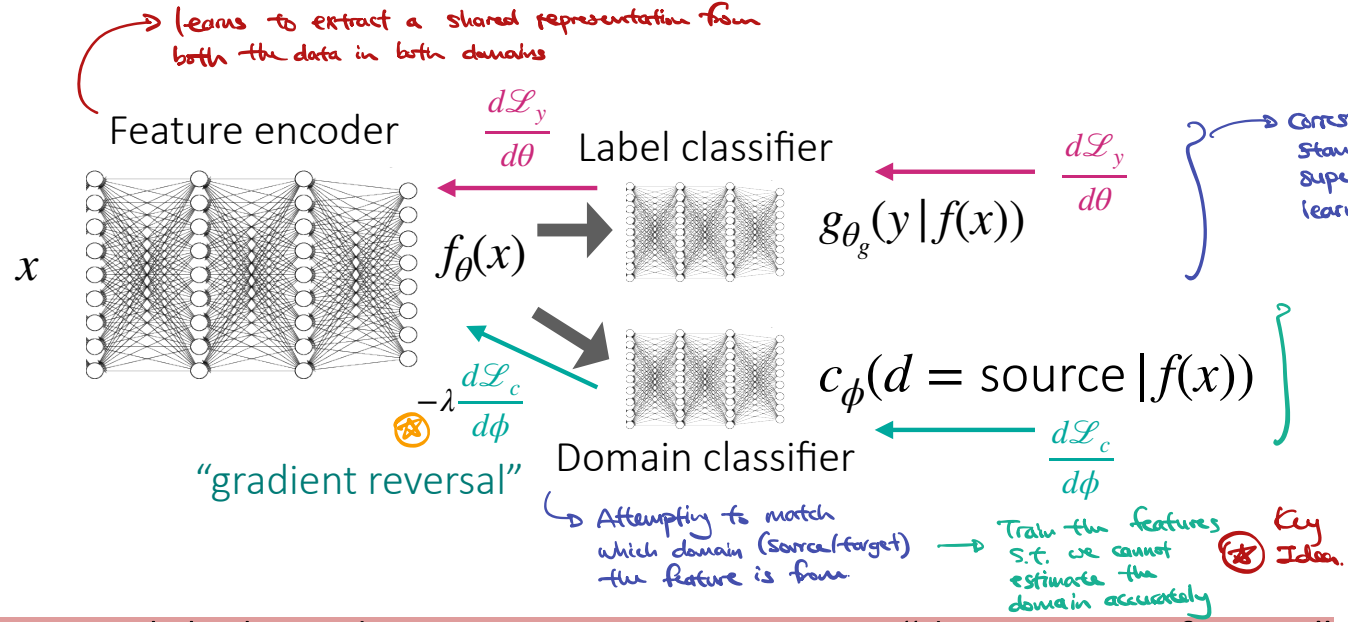
↳ Adversarial idea.

If samples are indistinguishable to discriminator, then distributions are the same.

Domain adaptation via feature alignment

Adversarial domain adaptation

Key idea: Try to fool a domain classifier $c(d = \text{source} \,|\, f(x))$.



Minimize label prediction error & maximize "domain confusion"

Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Algorithm

1. Update domain classifier G_ϕ wrt. $\mathcal{L}_c$
2. Update features $f(x)$ & label classifier g

In practice, iterate btw. these two stages

wrt $\mathcal{L}_y - \lambda \mathcal{L}_c$

λ: control var. to adjust the balance btw. the two.

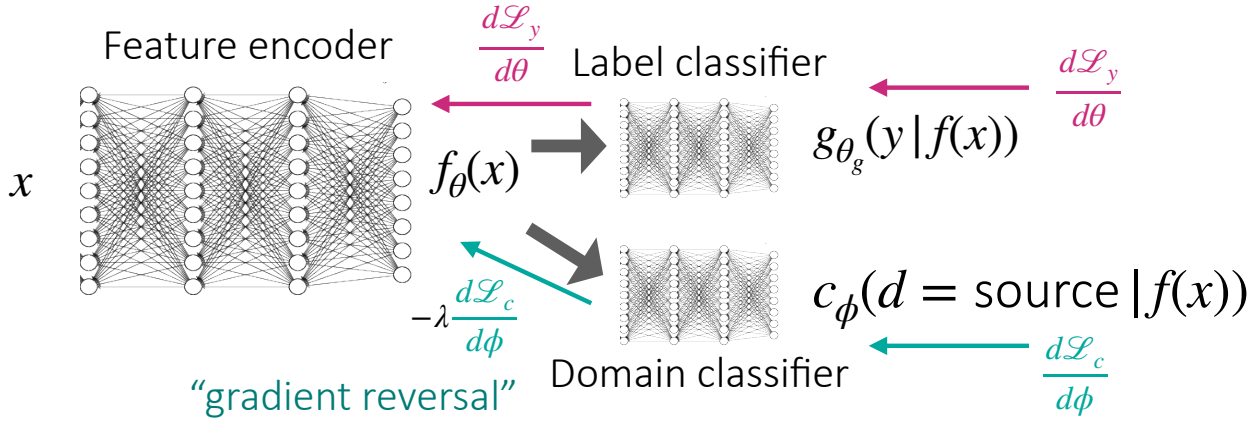
$\mathcal{L}_y$: D_S

$\mathcal{L}_c$: $D_S \cup D_T$ → evaluated using both the target & source dataset.

$\mathcal{L}_y$ → Label classifier loss

$\mathcal{L}_c$ → Domain classifier loss

Domain adaptation via feature alignment



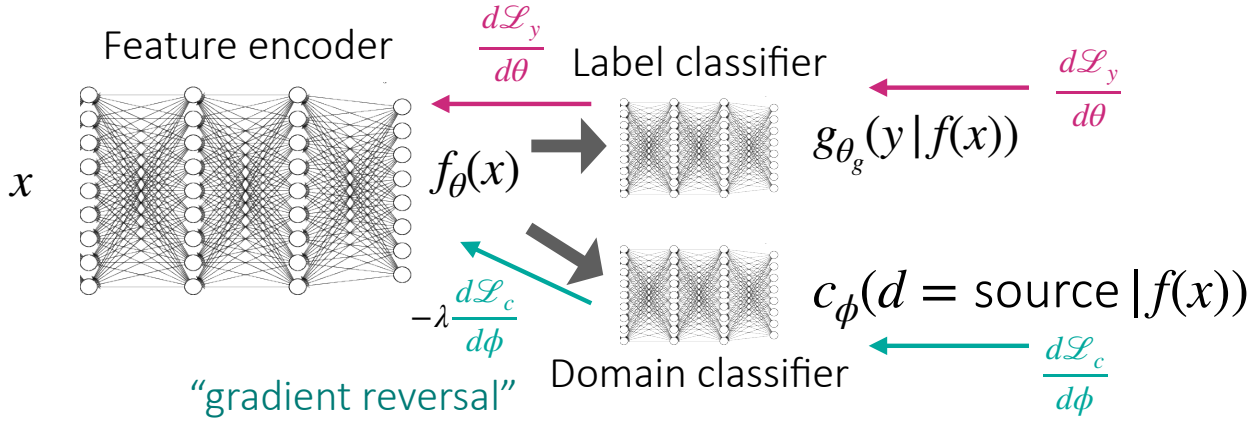
Full algorithm:

1. Randomly initialize encoder(s) $f_{\theta'}$ label classifier $g_{\theta_g'}$ domain classifier c_ϕ
2. Update domain classifier: $\min_\phi \mathscr{L}_c = -\mathbb{E}_{x\sim D_S}[\log c_\phi(f(x))] - \mathbb{E}_{x\sim D_T}[1 - \log c_\phi(f(x))].$
3. Update label classifier & encoder: $\min_{\theta,\theta_g} \mathbb{E}_{(x,y)\sim D_S}[\underbrace{L\left(g_{\theta_g}(f_\theta(x)), y\right)}_{\mathscr{L}_y \text{ Label classifier loss}}] - \underbrace{\lambda\mathscr{L}_c}_{\text{Domain classifier loss}}$
4. Repeat steps 2 & 3.

Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain adaptation via feature alignment



Can learn separate source and target encoder

Source encoder f_{θ_S} Target encoder f_{θ_T}

Make encoded samples $f_{\theta_S}(x), x \sim p_S(\cdot)$

indistinguishable from $f_{\theta_T}(x), x \sim p_T(\cdot)$

—> can give model more flexibility

↳ Do this if source ≠ target images look very different. → can possibly give the model too much flexibility

Different forms of domain adversarial training.

Option 1: Maximize domain classifier loss (gradient reversal, same as GANs)

↳ prev. slide

↳ predicting confidently the incorrect domain.

↓

this would give you features that can still distinguish btw source ≠ target

∴

alt. approach (somewhat prevents the above issue)

Option 2: Optimize for 50/50 guessing

↳ s.t. the domain classifier ends up outputting 50:50 guess

↳ Optimize it s.t. the classifier has no idea what the correct domain is.

Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

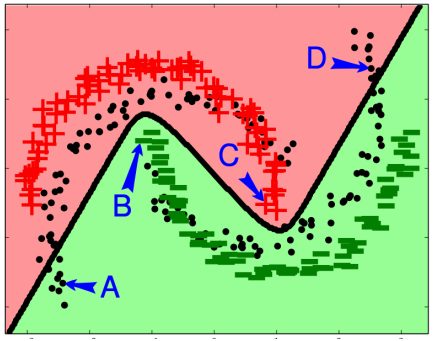
Domain adaptation via feature alignment

After domain adaptation!

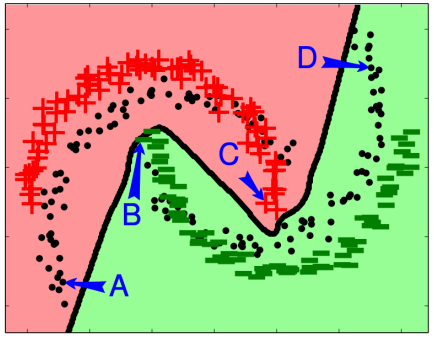
Toy example

source domain: +, —

target domain data: •



standard NN training

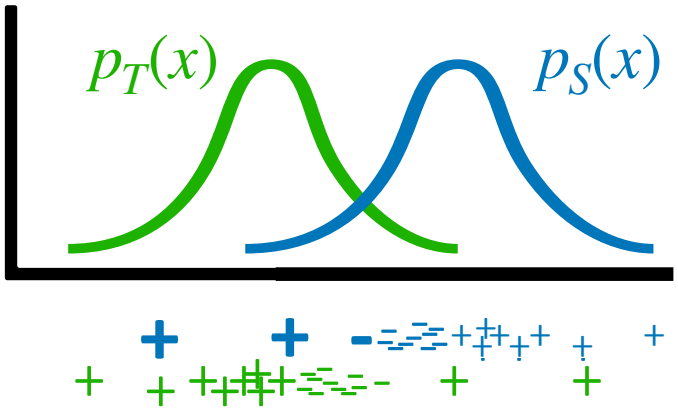


domain adversarial training

Method	Source	MNIST	Syn Numbers	SVHN	Syn Signs
	Target	MNIST-M	SVHN	MNIST	GTSRB
Source only		.5225	.8674	.5490	.7900
DANN		**.7666** (52.9%)	**.9109** (79.7%)	**.7385** (42.6%)	**.8865** (46.4%)
Train on target		.9596	.9220	.9942	.9980

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Importance weighting

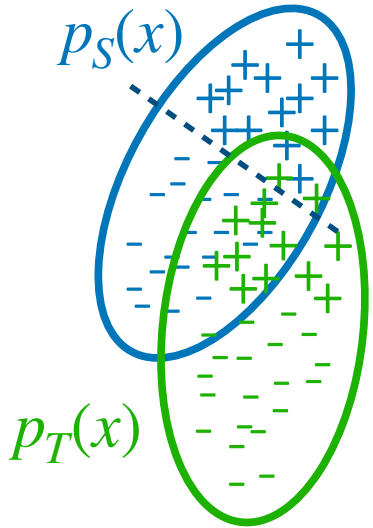


$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

\+ simple, can work well

— requires source distr. to cover target

Feature alignment



\+ fairly simple to implement, can work quite well

\+ doesn't require source data coverage

— involves adversarial optimization

— requires clear alignment in data

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - **Domain translation**

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

What if it is hard to align features?

Idea: translate between domains

i.e. $F : X_S \to X_T$ or $G : X_T \to X_S$

If you could translate source examples to target examples: Source → target

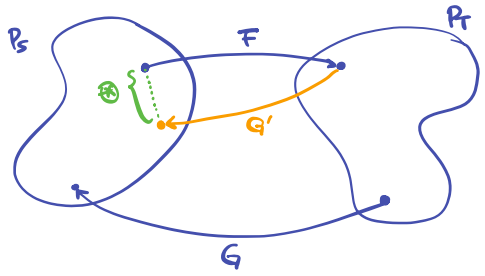
1. Translate labeled source dataset to target domain with F.
2. Train predictor on translated dataset.
3. Deploy predictor.

Alternatively, if you could translate from target to source: target → source

1. Train predictor on source dataset.
2. Translate target example to source domain with G.
3. Evaluate predictor on translated example.

(P) Will be operating on the original input space x, rather than the feature space!

Key question: How to translate between domains?



Domain Translation with CycleGAN

Idea: translate between domains

i.e. $F: X_S \rightarrow X_T$ or $G: X_T \rightarrow X_S$

Key question: How to translate between domains?

Step 1: Train F to generate images from $p_T(x)$ Source → target

and G to generate images from $p_S(x)$ Target → Source

Using GAN objective: $\mathcal{L}_{\text{GAN}} = \mathbb{E}_{x \sim p_T(\cdot)}[\log D_T(x)] + \mathbb{E}_{x \sim p_S(\cdot)}[1 - \log D_T(F(x))]$

↳ Train a classifier to determine the origin (source or target) → discriminator
+ a generator. Goal: Train the generator well enough s.t. the discriminator can't tell apart

i.e. it won't learn to map in a way that's consistent between the two domains

Challenge: The mapping is underconstrained, can be arbitrary.

Can we encourage models to learn a consistent, bijective mapping?

⊗ See orange arrow G'
⇒ Example of inconsistent mapping

Add another constraint

Step 2: Train F and G to be cyclically consistent.

$F(G(x)) \approx x$ and $G(F(x)) \approx x$ → Try to get ⊗ to be small

Domain Translation with CycleGAN

Idea: translate between domains

i.e. $F : X_S \rightarrow X_T$ or $G : X_T \rightarrow X_S$

Step 1: Train F to generate images from $p_T(x)$

and G to generate images from $p_S(x)$

Using GAN objective: $\mathcal{L}_{\text{GAN}} = \mathbb{E}_{x \sim p_T(\cdot)}[\log D_T(x)] + \mathbb{E}_{x \sim p_S(\cdot)}[1 - \log D_T(F(x))]$

Step 2: Train F and G to be cyclically consistent.

$F(G(x)) \approx x$ and $G(F(x)) \approx x$

i.e. $\mathbb{E}_{x \sim p_S(\cdot)} \|G(F(x)) - x\|_1 + \mathbb{E}_{x \sim p_T(\cdot)} \|F(G(x)) - x\|_1 = \mathcal{L}_{\text{cyc}}$

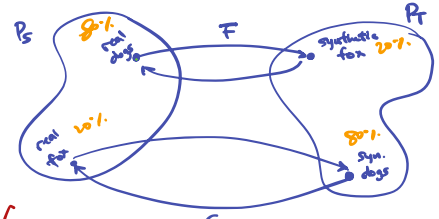
Full objective: $\mathcal{L}_{\text{GAN}}(F, D_T) + \mathcal{L}_{\text{GAN}}(G, D_S) + \lambda \mathcal{L}_{\text{cyc}}(F, G)$

Train to generate examples to look like target ex

make them gens look like source

Additional regularizer to ensure cycle consistency → when you do the fwd + inverse mapping, make it go back to where it came from.

Zhu, Park, Isola, Efros. CycleGAN. ICCV 2017



Domain Translation with CycleGAN

Don't even need labeled examples from our source dataset

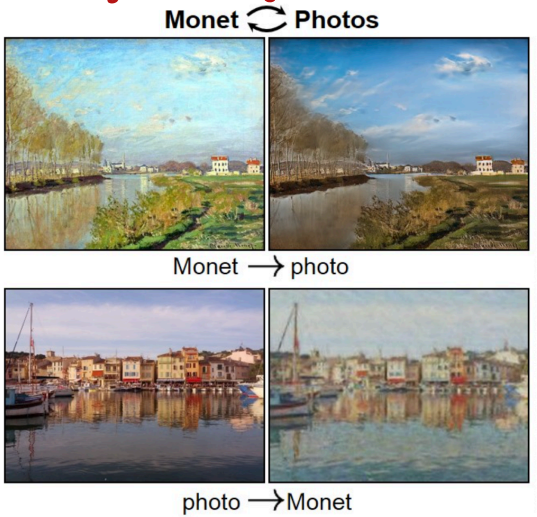
⇒ Purely unsupervised approach for mapping btw two domains.

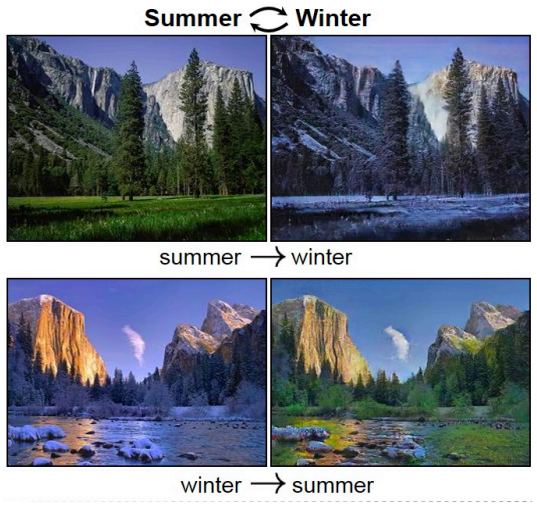
Idea: translate between domains

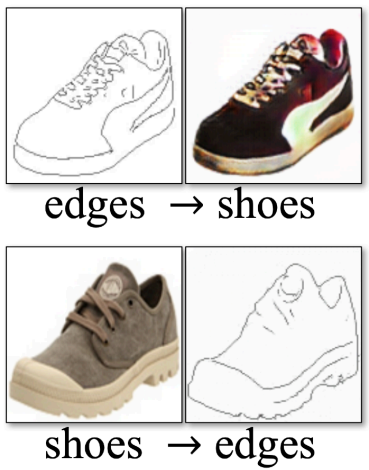
i.e. $F : X_S \rightarrow X_T$ or $G : X_T \rightarrow X_S$

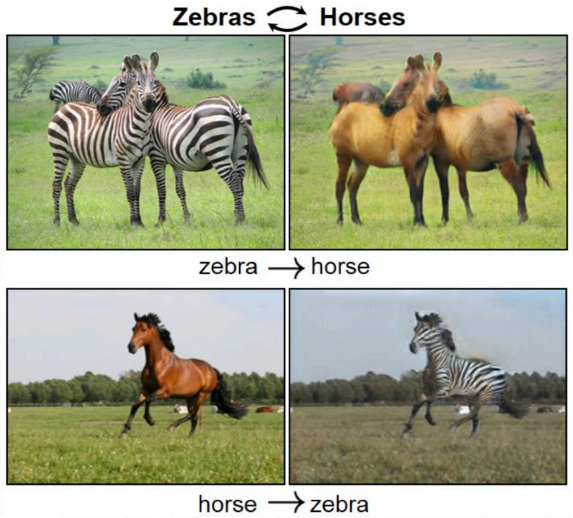
↳ This could happen, but a few things discourage this:
- the architecture could be constrained to change only the local features of the image.
- relative frequency of datasets will additionally encourage correct mappings
(eg) 80% real dogs, 20% real fox, 80% syn dogs, 20% syn fox)

↳ hard to make a dog out of a fox







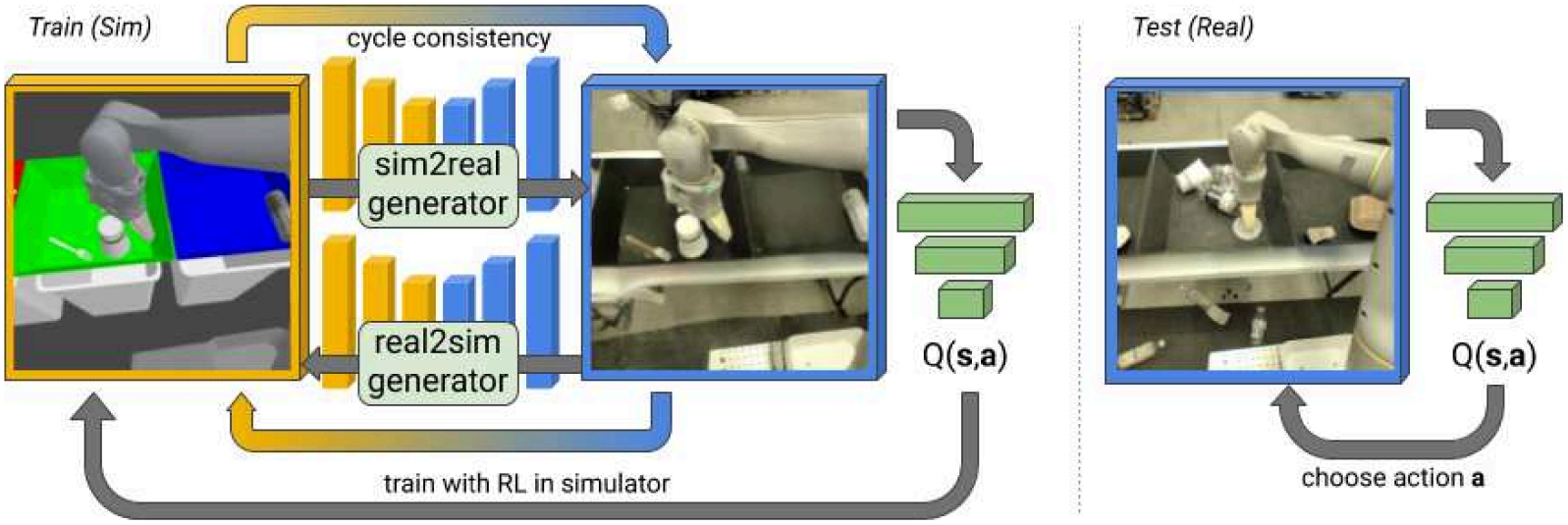


Zhu, Park, Isola, Efros. CycleGAN. ICCV 2017

CycleGAN for Domain Adaptation

Translate btw simulated robots and real robots

Robotics sim2real policy adaptation



"Domain Randomization": Try to randomize the simulator as much as possible

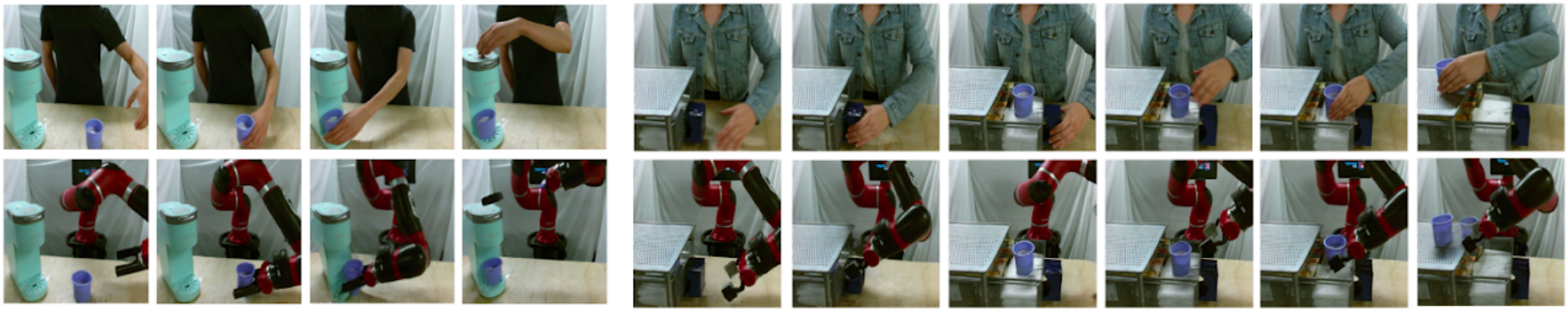
Simulation-to-Real Model	Robot 1 Grasp Success
Sim-Only [19]	21%
Randomized Sim [19]	37%
GAN	29%
CycleGAN	61%
GraspGAN	63%
RL-CycleGAN	**70%**

Rao, Harris, Irpan, Levine, Ibarz, Khansari. RL-CycleGAN. CVPR 2020

CycleGAN for Domain Adaptation

Human-robot domain adaptation

Input human images

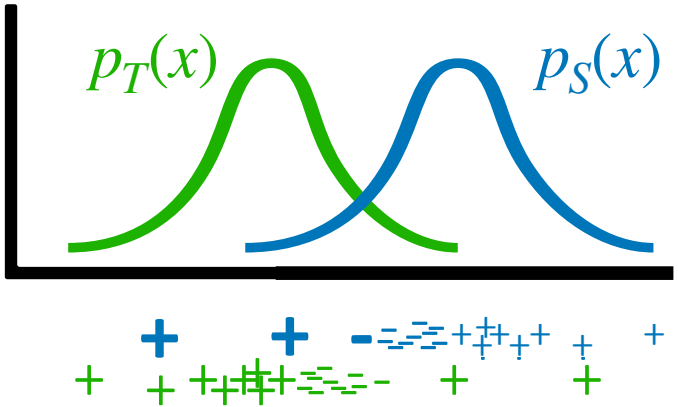


Generated images in robot domain

Easier to use this data directly

Smith, Dhawan, Zhang, Abbeel, Levine. RSS 2020

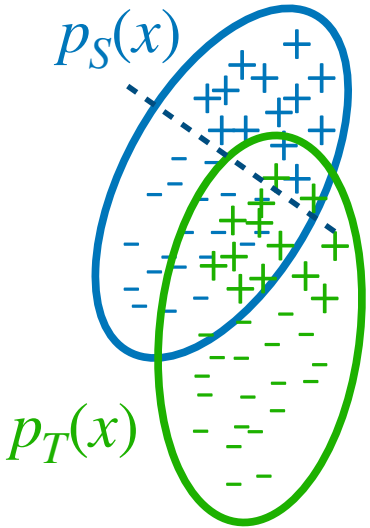
Importance weighting



$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

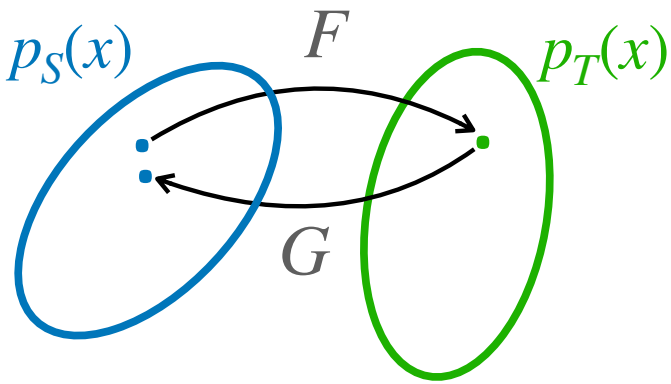
+ simple, can work well
— requires source distr. to cover target

Feature alignment



+ fairly simple to implement, can work quite well
+ doesn't require source coverage
— involves adversarial optimization
— requires clear alignment in data

Domain translation



+ conceptually neat, can work quite well
+ interpretable (easier to debug, cool pictures)
-- involves generative modeling & adversarial optimization
-- requires clear alignment in data

Allows you to work w/ more complex data

Domain-Adversarial Neural Networks

Combine the two approaches!

CycleGAN & DANN for Domain Adaptation

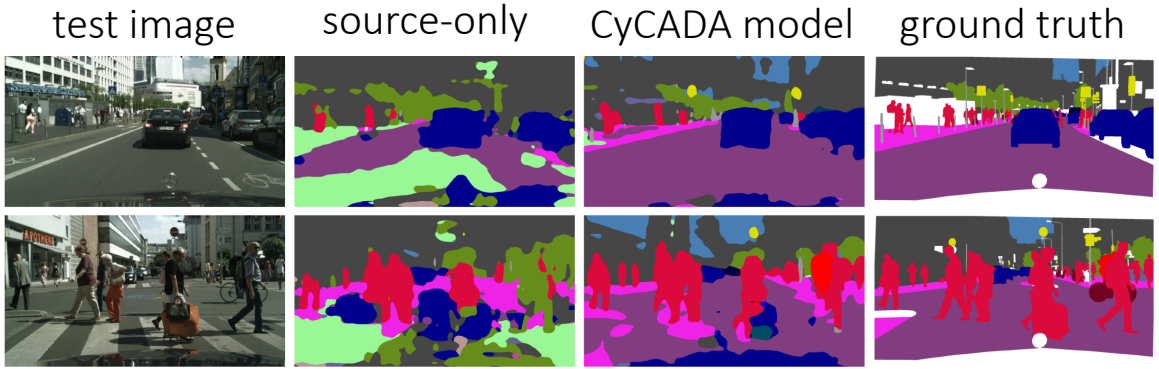
CyCADA: incorporates both cycle consistency & domain adversarial training

Synthetic driving data ↕ Real driving data (Cityscapes)

⇒ Much more accurate segmentation!

Character recognition

Model	USPS → MNIST	SVHN → MNIST
Source only	69.6 ± 3.8	67.1 ± 0.6
DANN (Ganin et al., 2016)	-	73.6
DTN (Taigman et al., 2017a)	-	84.4
CoGAN (Liu & Tuzel, 2016b)	89.1	-
ADDA (Tzeng et al., 2017)	90.1 ± 0.8	76.0 ± 1.8
PixelDA (Bousmalis et al., 2017b)	-	-
UNIT (Liu et al., 2017)	93.6	**90.5***
CyCADA (Ours)	**96.5 ± 0.1**	**90.4 ± 0.4**
Target Fully Supervised	99.2 ± 0.1	99.2 ±0.1



GTA5 → Cityscapes

	Architecture	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorbike	bicycle	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	A	26.0	14.9	65.1	5.5	12.9	8.9	6.0	2.5	70.0	2.9	47.0	24.5	0.0	40.0	12.1	1.5	0.0	0.0	0.0	17.9	41.9	54.0
FCN-wld (Hoffman et al., 2016)	A	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1	-	-
CDA (Zhang et al., 2017b)	A	26.4	22.0	74.7	6.0	11.9	8.4	16.3	11.1	75.7	13.3	**66.5**	38.0	**9.3**	55.2	**18.8**	18.9	0.0	**16.8**	**14.6**	27.8	-	-
FCTN (Zhang et al., 2017a)	A	72.2	28.4	74.9	18.3	10.8	**24.0**	**25.3**	17.9	80.1	**36.7**	61.1	44.7	0.0	74.5	8.9	1.5	0.0	0.0	0.0	30.5	-	-
CyCADA (Ours)	A	**85.2**	**37.2**	**76.5**	**21.8**	**15.0**	23.8	22.9	**21.5**	**80.5**	31.3	60.7	**50.5**	9.0	**76.9**	17.1	**28.2**	**4.5**	9.8	0.0	**35.4**	**73.8**	**83.6**
Oracle - Target Supervised	A	96.4	74.5	87.1	35.3	37.8	36.4	46.9	60.1	89.0	54.3	89.8	65.6	35.9	89.4	38.6	64.1	38.6	40.5	65.1	60.3	87.6	93.1
Source only	B	42.7	26.3	51.7	5.5	6.8	13.8	23.6	6.9	75.5	11.5	36.8	49.3	0.9	46.7	3.4	5.0	0.0	5.0	1.4	21.7	47.4	62.5
CyCADA (Ours)	B	**79.1**	**33.1**	**77.9**	**23.4**	**17.3**	**32.1**	**33.3**	31.8	**81.5**	**26.7**	**69.0**	**62.8**	14.7	**74.5**	**20.9**	**25.6**	**6.9**	**18.8**	20.4	**39.5**	**72.4**	**82.3**
Oracle - Target Supervised	B	97.3	79.8	88.6	32.5	48.2	56.3	63.6	73.3	89.0	58.9	93.0	78.2	55.2	92.2	45.0	67.3	39.6	49.9	73.6	67.4	89.6	94.3

Table 4: Adaptation between GTA5 and Cityscapes, showing IoU for each class and mean IoU, freq-weighted IoU and pixel accuracy. CyCADA significantly outperforms baselines, nearly closing the gap to the target-trained oracle on pixel accuracy.

Hoffman et al. ICML 2018

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

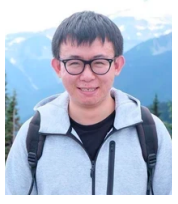
Course Reminders

Optional homework 4 due next **Monday**.

Project milestone due next **Wednesday**

Azure: If you are close to running out of credits, proactively request more in private Ed post.

Next time: Domain generalization



by Huaxiu Yao