6.7900 Machine learning (Fall 2024)

Lecture 15: unsupervised domain adaptation

(supporting slides)

Domain shifts

- The training (source) data is often not exactly like the test (target) data
- The differences arise for many reasons, and may be known, partially known, or unknown

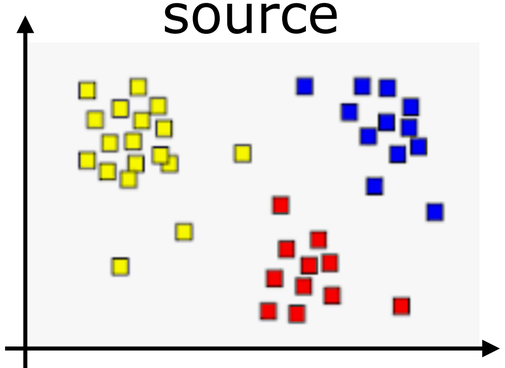
Domain shifts

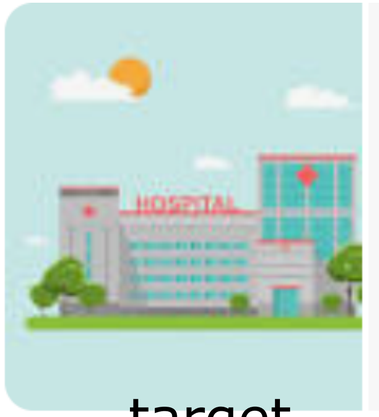
- The training (source) data is often not exactly like the test (target) data
- The differences arise for many reasons, and may be known, partially known, or unknown
- **E.g., temporal changes**
  - predictors trained from older data, applied to new cases (such as customer behavior)

Domain shifts

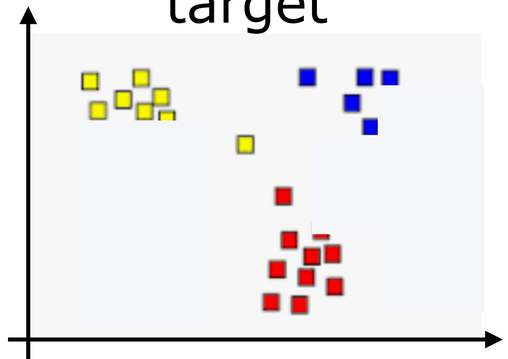
- The training (source) data is often not exactly like the test (target) data
- The differences arise for many reasons, and may be known, partially known, or unknown
- **E.g., demographic/context changes**
  - speech recognition system trained with one set of speakers, applied more broadly
  - risk predictors that are based on patient data from one hospital but used in a different hospital with different demographics

source



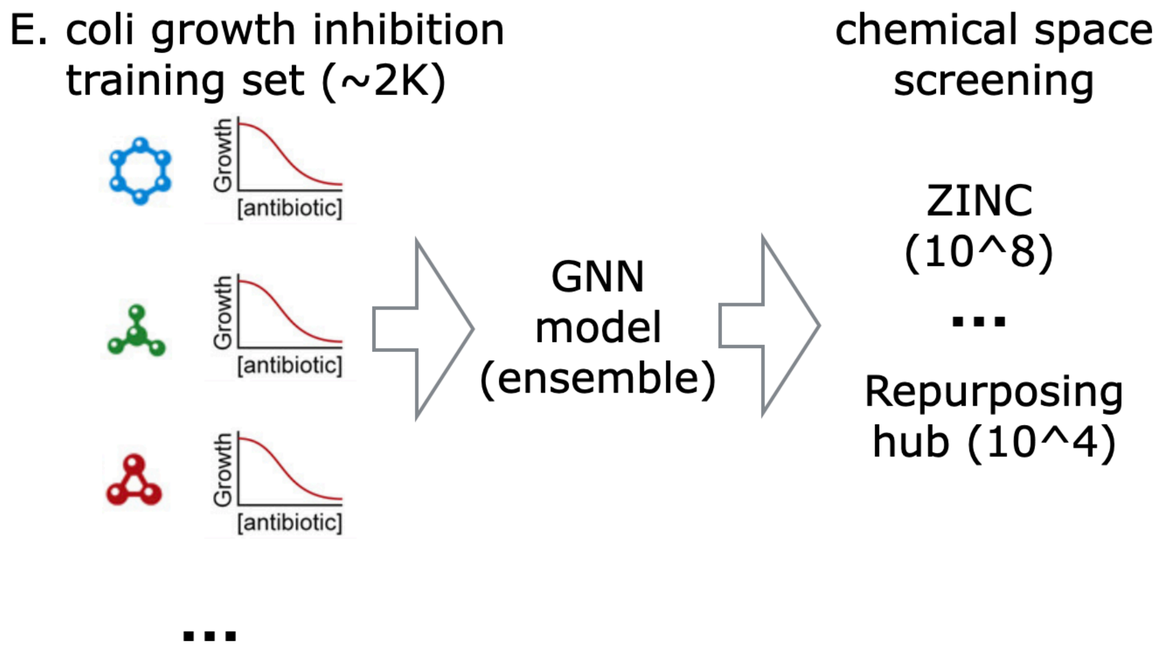


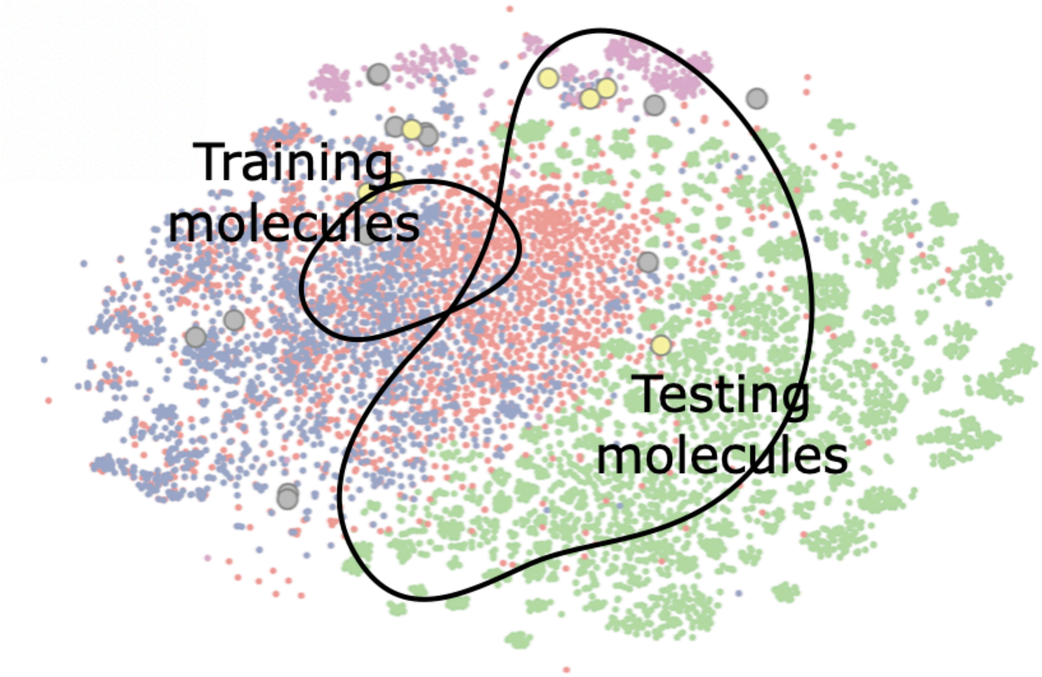
target



Domain shifts

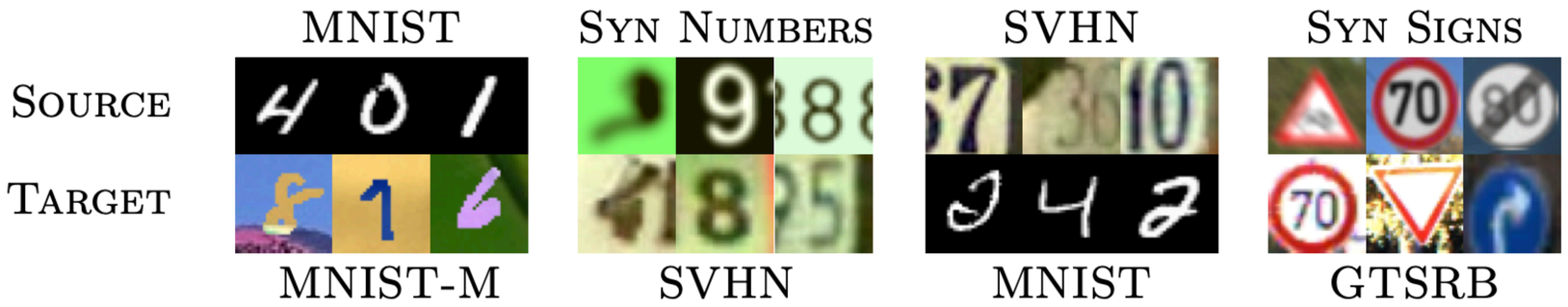
- The training (source) data is often not exactly like the test (target) data
- The differences arise for many reasons, and may be known, partially known, or unknown
- **E.g., actions we take**
  - predictors that are used in iterative experiment design (e.g., molecule optimization) where the current data comes from earlier rounds of s





Can we just train on source, test on target?

- Performance can be substantially affected by “domain shift”… we need ways of dealing with such shifts



Method	Source	MNIST	Syn Numbers	SVHN	Syn Signs
	Target	MNIST-M	SVHN	MNIST	GTSRB
Source only		.5225	.8674	.5490	.7900
Train on target		.9596	.9220	.9942	.9980

test accuracy

[Ganin et al. 2015]

Domain adaptation possibilities

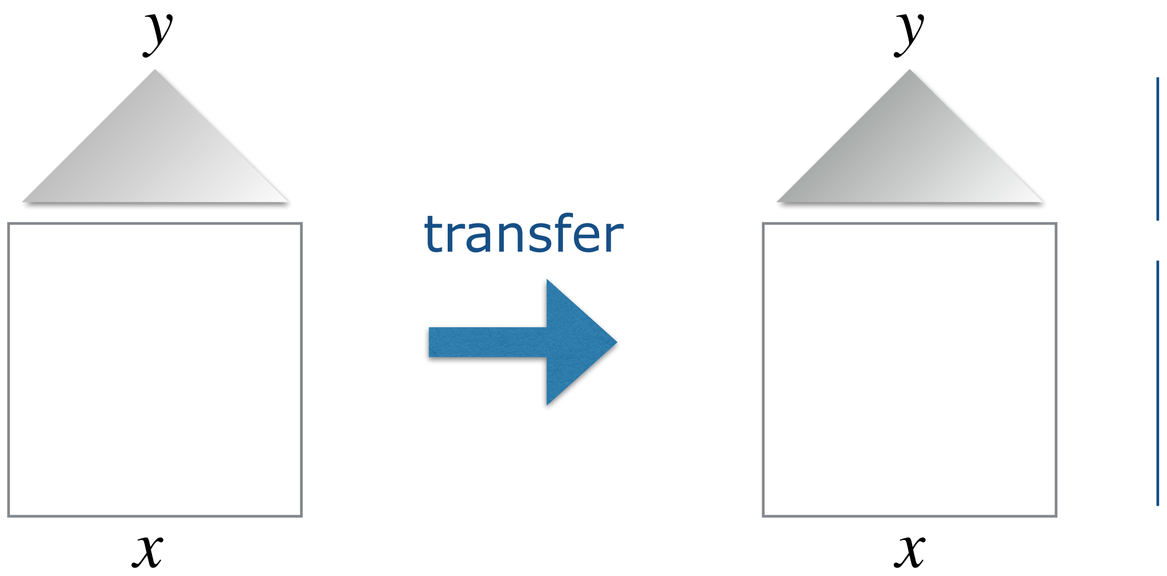
- Our goal is to learn from labeled/annotated training (source) examples and transfer this knowledge for use in a different — target — domain

- Supervised domain adaptation
  - we have some annotated examples also from the target domain
- Semi-supervised domain adaptation
  - we have lots of unlabeled target examples and just a few annotations
- **Unsupervised domain adaptation**
  - at best we only have unlabeled examples from the target domain

Domain adaptation: unsupervised

Source domain,
lots of labeled examples

Target domain,
only unlabeled examples



We assume the labels are still the same (e.g., long neck?)

water birds

land birds

Adjusting to three different types of shifts

- Often we can make some assumptions about the type of changes between the domains:

- **Case #1 (label shift only):** the proportions of labels we wish to predict differs from the source to the target domain, the data are otherwise generated the same

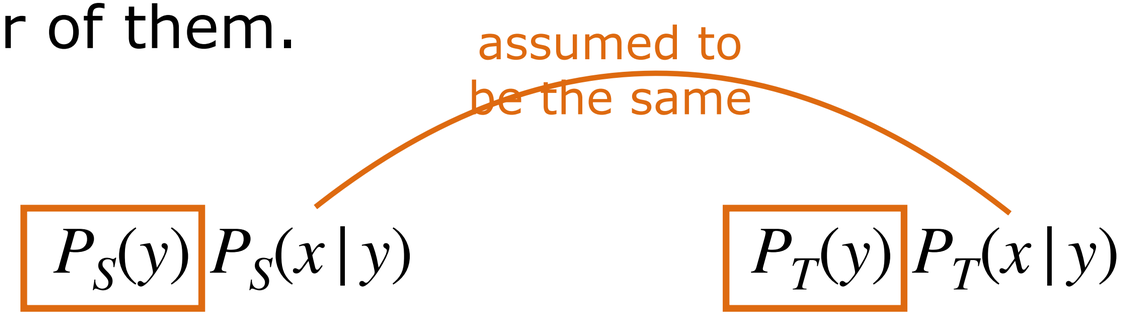
- **Case #2 (co-variate shift only):** the distribution of co-variates (distribution over x's) differs from the source to the target domain, the data are otherwise generated the same

- **Case #3 (co-variate transformation only):** the target domain co-variates are "evolved" or transformed versions of the source co-variates; relation to labels remains after the transformation

- (these are not mutually separable, and the list is not complete)

Label shift

- The problem is how we can adjust an already trained classifier $P_S(y|x)$ so that it is appropriate for the target domain where the label proportions are different
- E.g., a new specialized clinic appears next to a hospital. So patients with certain illnesses covered by the clinic are now less frequently coming to the hospital. But we can hypothesize that those who still come "look" the same as before, there are just fewer of them.



we have trained $P_S(y|x)$

$P_T(y|x) = ?$

what is the adjusted classifier we should run on target examples?

= known to us

Label shift

- The problem is how we can adjust an already trained classifier $P_S(y|x)$ so that it is appropriate for the target domain where the label proportions are different
- Let's consider any x (fixed, only y varies) and try to construct the target classifier based on the available information

$$P_T(y|x) \propto P_T(x|y)P_T(y)$$

$$= P_S(x|y)P_T(y)$$ class conditional distributions assumed to be the same

$$= P_S(x|y)P_S(y)\frac{P_T(y)}{P_S(y)}$$

$$\propto \frac{P_S(x|y)P_S(y)}{P_S(x)}\frac{P_T(y)}{P_S(y)}$$ x is fixed, so we can include any x term

$$= P_S(y|x)\frac{P_T(y)}{P_S(y)}$$ we have expressed the target classifier proportionally to the source classifier and the label ratios; normalizing this across y gives our target classifier

Adjusting to three different types of shifts

- Often we can make some assumptions about the type of changes between the domains:

- **Case #1 (label shift only):** the proportions of labels we wish to predict differs from the source to the target domain, the data are otherwise generated the same

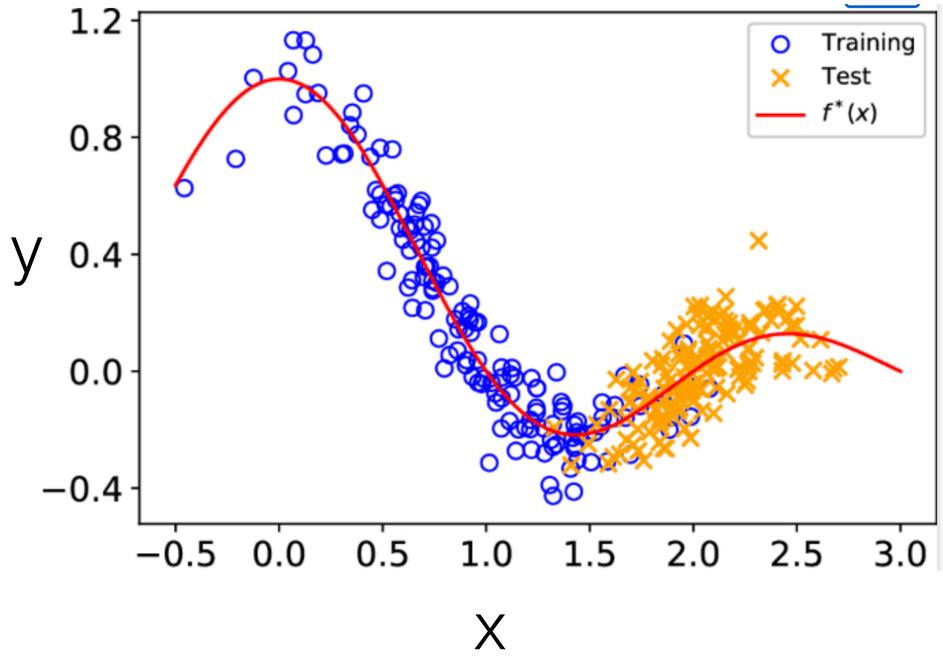
- **Case #2 (co-variate shift only):** the distribution of co-variates (distribution over x's) differs from the source to the target domain, the data are otherwise generated the same

- **Case #3 (co-variate transformation only):** the target domain co-variates are "evolved" or transformed versions of the source co-variates; relation to labels remains after the transformation

- (these are not mutually separable, and the list is not complete)

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)

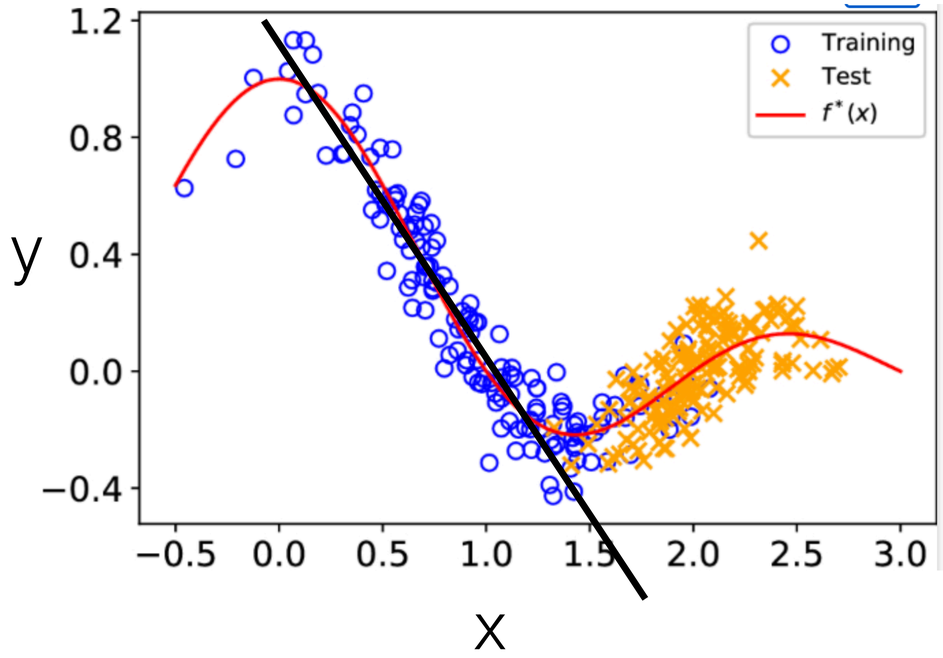


[image adapted from Zhang et al. 2021]

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)

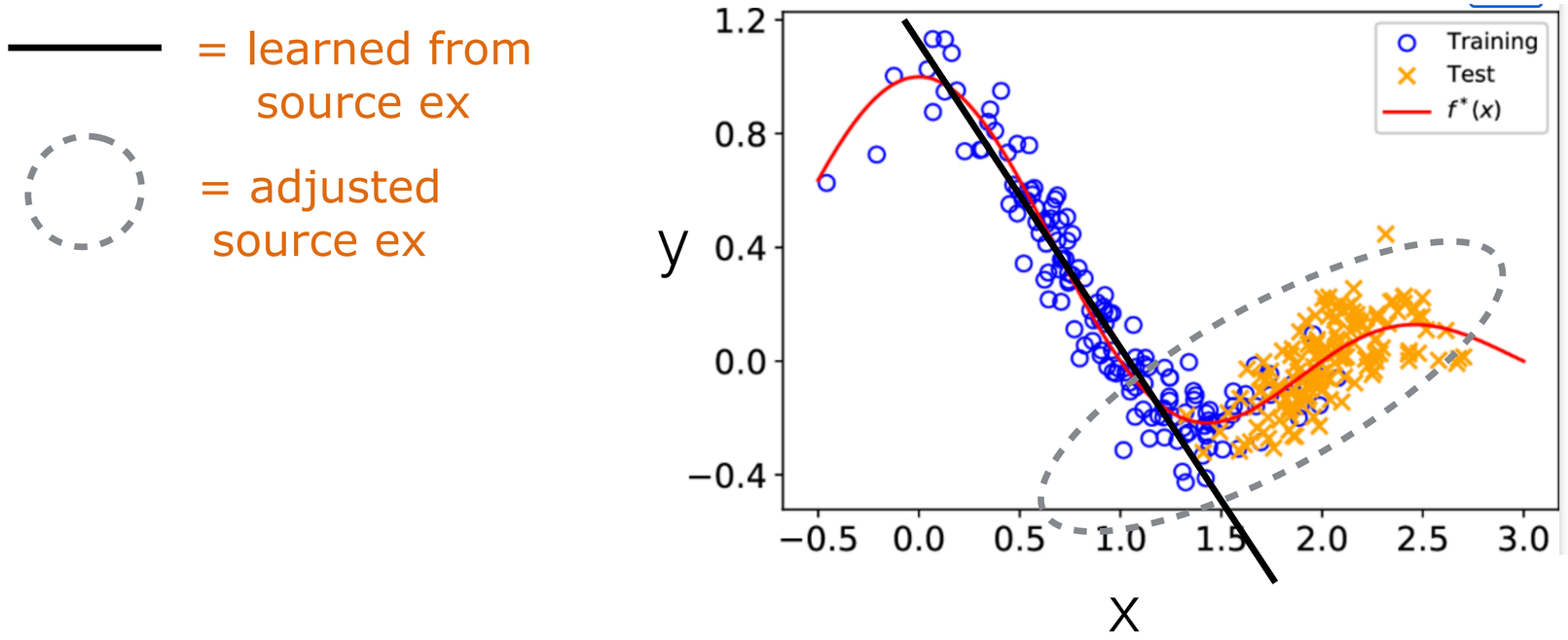
——— = learned from source ex



[image adapted from Zhang et al. 2021]

Co-variate shift

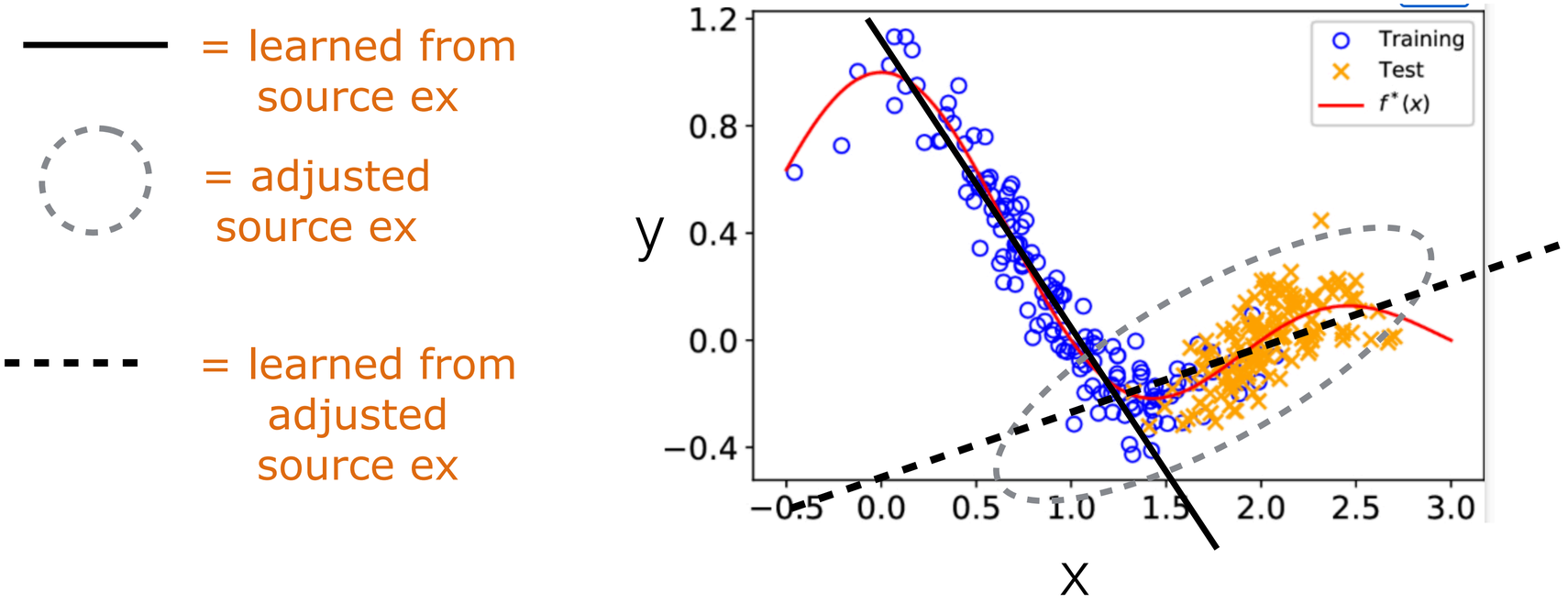
- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)



[image adapted from Zhang et al. 2021]

Co-variate shift

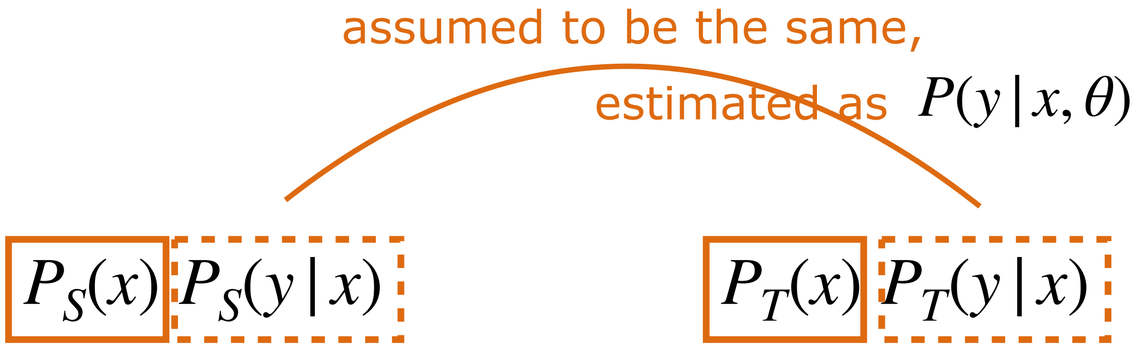
- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)



[image adapted from Zhang et al. 2021]

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)



= we have lots of samples from

= we wish to estimate

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)
- We would like to estimate θ that minimizes

$$\sum_{x,y} P_T(x,y)L(x,y,\theta)$$

average over the target distribution

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)
- We would like to estimate θ that minimizes

$$\sum_{x,y} P_T(x,y)L(x,y,\theta) = \sum_{x,y} P_S(x,y)\frac{P_T(x,y)}{P_S(x,y)}L(x,y,\theta)$$

average over the target distribution

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)
- We would like to estimate θ that minimizes

$$\sum_{x,y} P_T(x,y)L(x,y,\theta) = \sum_{x,y} P_S(x,y)\frac{P_T(x,y)}{P_S(x,y)}L(x,y,\theta)$$

average over the target distribution

$$= \sum_{x,y} P_S(x,y)\frac{P_T(x)P_T(y|x)}{P_S(x)P_S(y|x)}L(x,y,\theta)$$

Co-variate shift

- When $P_T(x)$ differs from $P_S(x)$ we could still in principle use the same classifier or regression model $P(y|x,\theta)$ trained from the source examples
- But estimating this regression model from the source data may improperly focus on source only examples (e.g., x's where $P_S(x)$ is large but $P_T(x)$ is small)
- We would like to estimate θ that minimizes

$$\sum_{x,y} P_T(x,y)L(x,y,\theta) = \sum_{x,y} P_S(x,y)\frac{P_T(x,y)}{P_S(x,y)}L(x,y,\theta)$$

average over the target distribution

$$= \sum_{x,y} P_S(x,y)\frac{P_T(x)P_T(y|x)}{P_S(x)P_S(y|x)}L(x,y,\theta)$$

$$= \sum_{x,y} P_S(x,y)\frac{P_T(x)}{P_S(x)}L(x,y,\theta)$$

weighted average over the source distribution (importance sampling)

Co-variate shift

- How do we get the ratio $P_T(x)/P_S(x)$ used as weights in the new estimation criterion? If x is high dimensional, estimating such distributions is challenging
- It turns out we can train a simple domain classifier to estimate this ratio
  - only use unlabeled examples from the source and target
  - label examples according to whether they come from the source S or target T
  - estimate a domain classifier $Q(S|x)$ from such labeled data

- If the domain classifier is "perfect" then

$$Q(S|x) = \frac{P_S(x)}{P_S(x) + P_T(x)} \qquad \frac{Q(T|x)}{Q(S|x)} = \frac{P_T(x)}{P_S(x)}$$

- So we can get the ratio weights required for estimation directly from the domain classifier

Co-variate shift: importance sampling

- Since we can now get the required probability ratios from the domain classifier, we can estimate

$$\sum_{x,y} P_T(x,y)L(x,y,\theta) = \sum_{x,y} P_S(x,y)\frac{P_T(x)}{P_S(x)}L(x,y,\theta)$$

$$= \sum_{x,y} P_S(x,y)\frac{Q(T\,|\,x)}{Q(S\,|\,x)}L(x,y,\theta)$$

Co-variate shift: importance sampling

- Since we can now get the required probability ratios from the domain classifier, we can estimate

$$\sum_{x,y} P_T(x,y)L(x,y,\theta) = \sum_{x,y} P_S(x,y)\frac{P_T(x)}{P_S(x)}L(x,y,\theta)$$

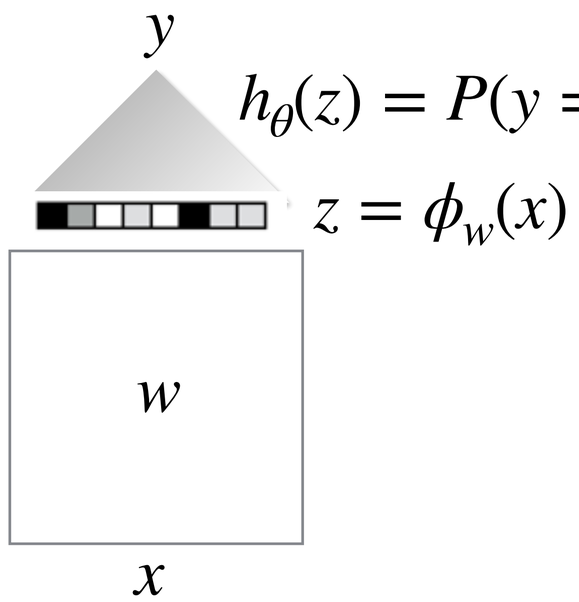
$$= \sum_{x,y} P_S(x,y)\frac{Q(T|x)}{Q(S|x)}L(x,y,\theta)$$

$$\approx \frac{1}{n}\sum_{i=1}^{n} \frac{Q(T|x^i)}{Q(S|x^i)}L(x,y,\theta)$$

- where the labeled examples are samples from the source domain. The classifier is estimated from unlabeled source and target samples (co-variates only)

Domain alignment

- We can adjust the intermediate representation of examples so as to make examples look the same on this level across domains (so no re-weighting needed!)

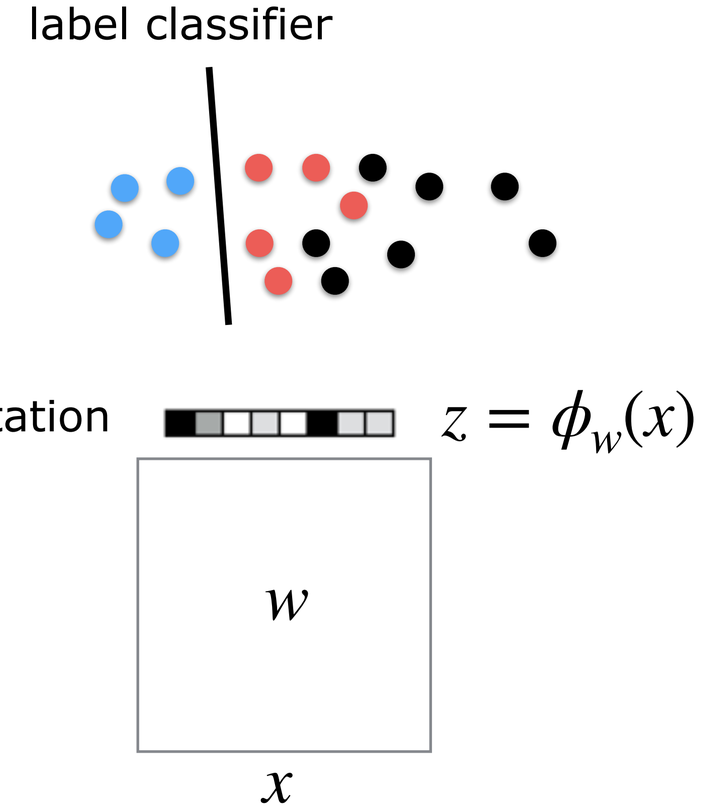


Method	Source / Target	MNIST / MNIST-M	Syn Numbers / SVHN	SVHN / MNIST	Syn Signs / GTSRB
Source only		.5225	.8674	.5490	.7900
Train on target		.9596	.9220	.9942	.9980

test

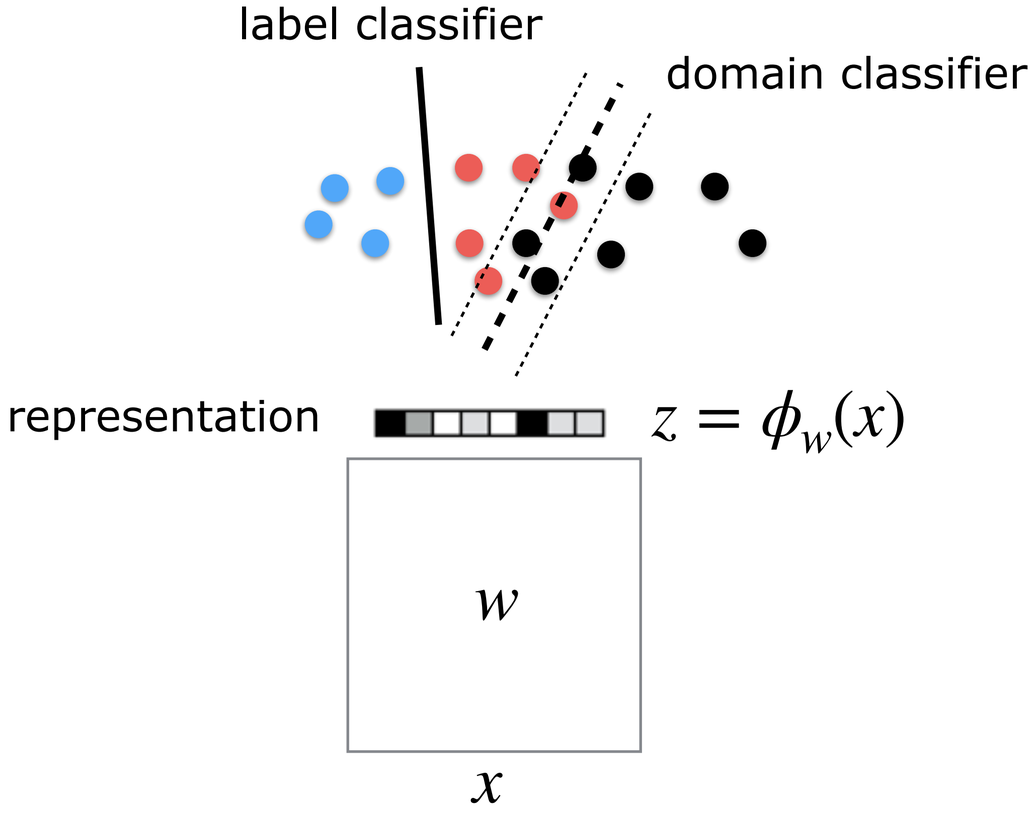
Adversarial domain alignment

- We would like to adjust the intermediate representation of examples so that
  1. the source labeled examples are easy to classify correctly
  2. the examples look the same across domains on that level



Adversarial domain alignment

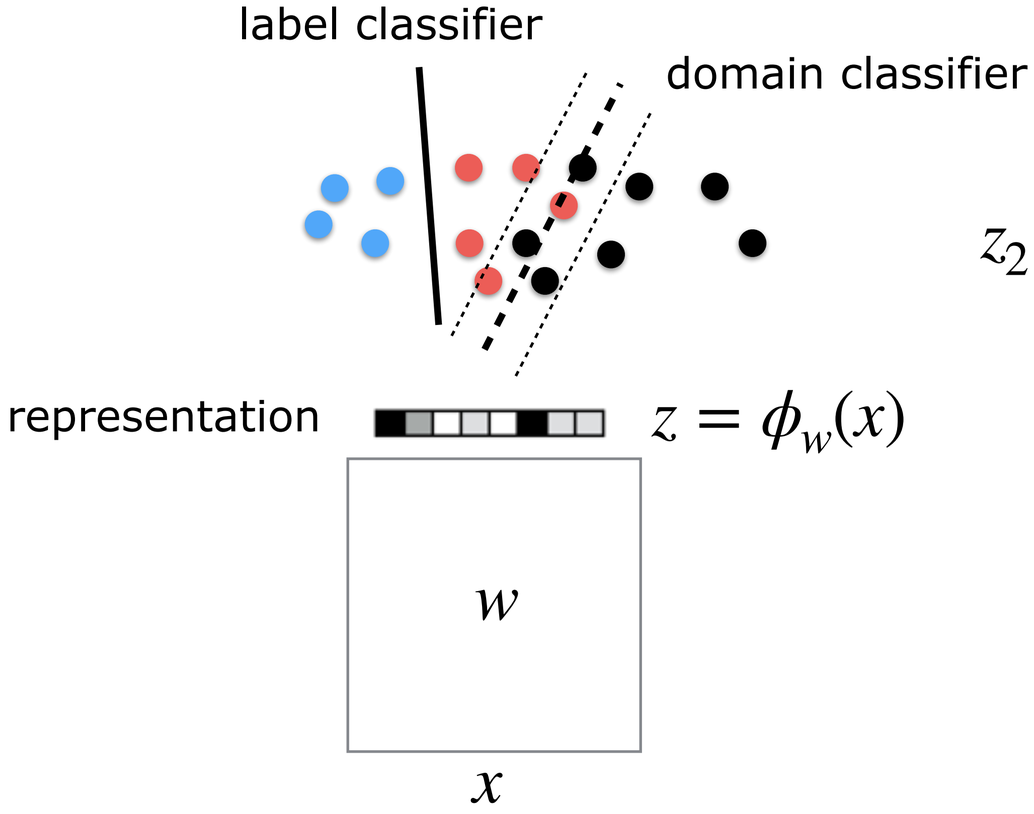
- We would like to adjust the intermediate representation of examples so that
  1. the source labeled examples are easy to classify correctly
  2. the examples look the same across domains on that level



used to highlight differences between source and target distributions; provides a training signal for the representation so as to align the source and the target

Adversarial domain alignment

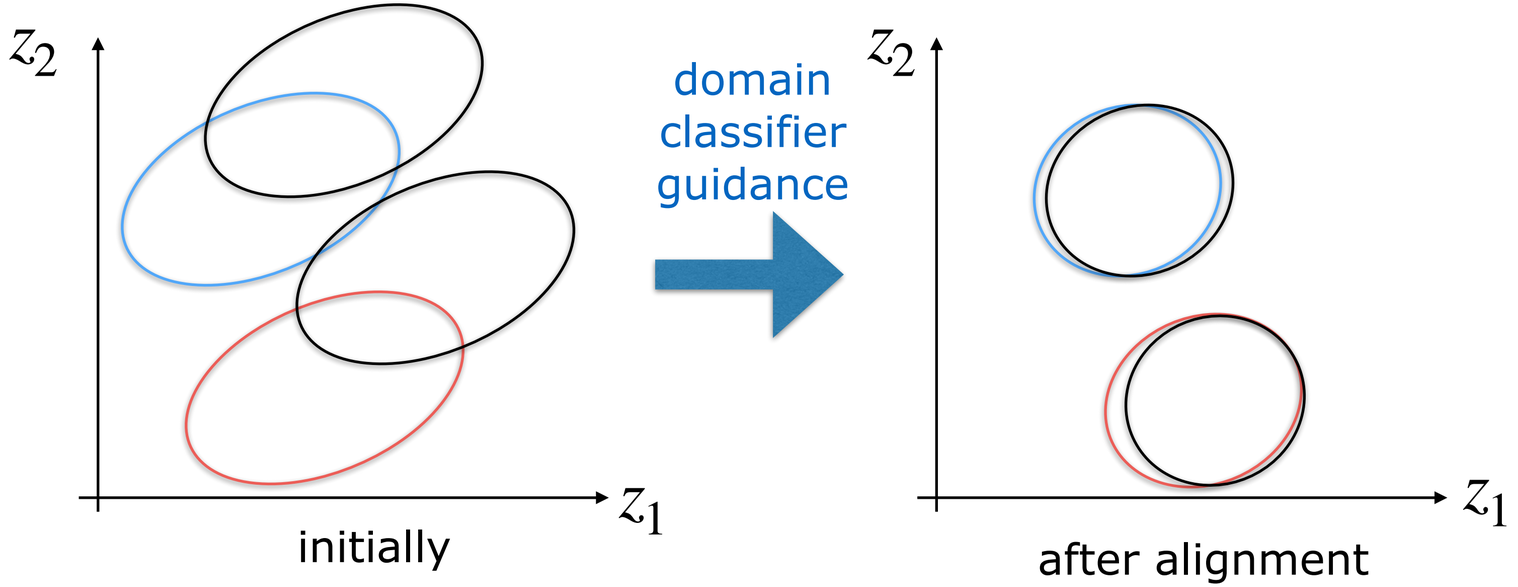
- We would like to adjust the intermediate representation of examples so that
  1. the source labeled examples are easy to classify correctly
  2. the examples look the same across domains on that level



labeled source domain examples

unlabeled target domain examples

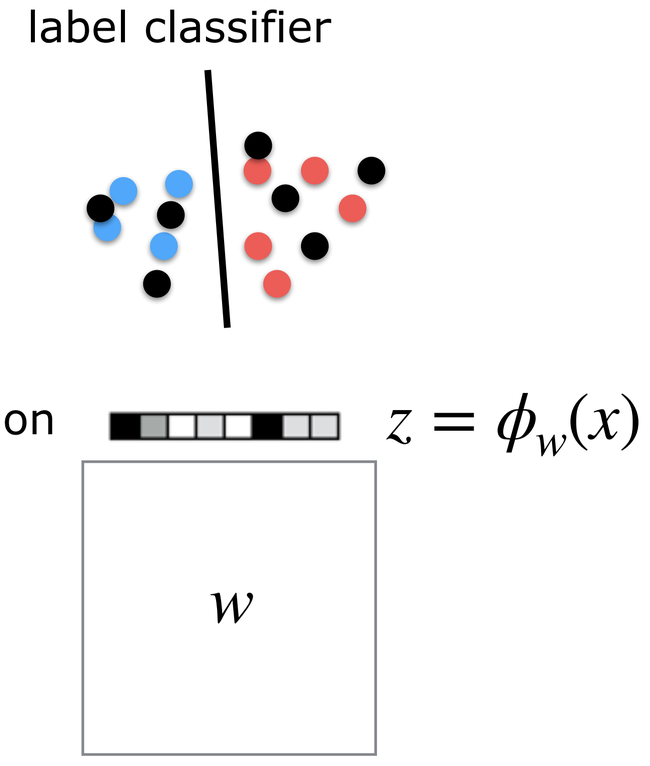
used to highlight differences between source and target distributions; provides a training signal for the representation so as to align the source and the target



Adversarial domain alignment: training

- We wish to find an intermediate representation $z = \phi_w(x)$ such that
  - (1) the source label classifier $h_\theta(\phi_w(x)) = P(y = 1 \,|\, \phi_w(x), \theta)$ is accurate
  - (2) source and target domain marginals look the same at the level of the representation

$$\tilde{\mathbb{P}}_S(z) = \mathbb{P}_S(z = \phi_w(x)) \approx \mathbb{P}_T(z = \phi_w(x)) = \tilde{\mathbb{P}}_T(z)$$



Adversarial domain alignment: training

- We wish to find an intermediate representation $z = \phi_w(x)$ such that
  - (1) the source label classifier $h_\theta(\phi_w(x)) = P(y = 1 \,|\, \phi_w(x), \theta)$ is accurate
  - (2) source and target domain marginals look the same at the level of the representation

$$\tilde{\mathbb{P}}_S(z) = \mathbb{P}_S(z = \phi_w(x)) \approx \mathbb{P}_T(z = \phi_w(x)) = \tilde{\mathbb{P}}_T(z)$$

- This can be cast as a regularization problem

$$\min_{\theta, w} \quad E_{(x,y)\sim\mathbb{P}_S} L(h_\theta(\phi_w(x)), y) + \lambda \, d(\tilde{\mathbb{P}}_S, \tilde{\mathbb{P}}_T)$$

- where $d(\mathbb{P}_S, \mathbb{P}_T)$ is a divergence measure between two distributions, here over induced feature vectors
- The divergence measure can be defined via a domain classifier that highlights how the two distributions differ; the goal of the representation is then to minimize this difference

Adversarial domain alignment: training

- We wish to find an intermediate representation $z = \phi_w(x)$ such that
  - (1) the source label classifier $h_\theta(\phi_w(x)) = P(y = 1 \,|\, \phi_w(x), \theta)$ is accurate
  - (2) source and target domain marginals look the same at the level of the representation

$$\tilde{\mathbb{P}}_S(z) = \mathbb{P}_S(z = \phi_w(x)) \approx \mathbb{P}_T(z = \phi_w(x)) = \tilde{\mathbb{P}}_T(z)$$

- This can be cast as a regularization problem

depends on w

$$\min_{\theta, w} \quad E_{(x,y)\sim\mathbb{P}_S} L(h_\theta(\phi_w(x)), y) + \lambda\, d(\tilde{\mathbb{P}}_S, \tilde{\mathbb{P}}_T)$$

depends on w

- where $d(\mathbb{P}_S, \mathbb{P}_T)$ is a divergence measure between two distributions, here over induced feature vectors
- The divergence measure can be defined via a domain classifier that highlights how the two distributions differ; the goal of the representation is then to minimize this difference

Domain adversarial training

- For example (Jensen-Shannon divergence):

$$d(\tilde{\mathbb{P}}_S, \tilde{\mathbb{P}}_T) = 2\,JSD(\tilde{\mathbb{P}}_T, \tilde{\mathbb{P}}_S) - \log(4) =$$

$$= \max_{\beta} \left\{ E_{z\sim\tilde{\mathbb{P}}_T} \log Q(T\,|\,z,\beta) + E_{z\sim\tilde{\mathbb{P}}_S} \log Q(S\,|\,z,\beta) \right\}$$

log-likelihood of correctly classifying the domain label

$$= \max_{\beta} \left\{ E_{x\sim\mathbb{P}_T} \log Q(T\,|\,\phi_w(x),\beta) + E_{x\sim\mathbb{P}_S} \log Q(S\,|\,\phi_w(x),\beta) \right\}$$

- The domain classifier is the “adversary” that aims to find differences between the domains at the level of $z = \phi_w(x)$

Domain adversarial training: three players

- (1) Adversary updates domain predictions (**max** divergence)

$$\beta = \beta + \eta \nabla_\beta \left\{ E_{x \sim \mathbb{P}_T} \log Q(T \mid \phi_w(x), \beta) + E_{x \sim \mathbb{P}_S} \log Q(S \mid \phi_w(x), \beta) \right\}$$

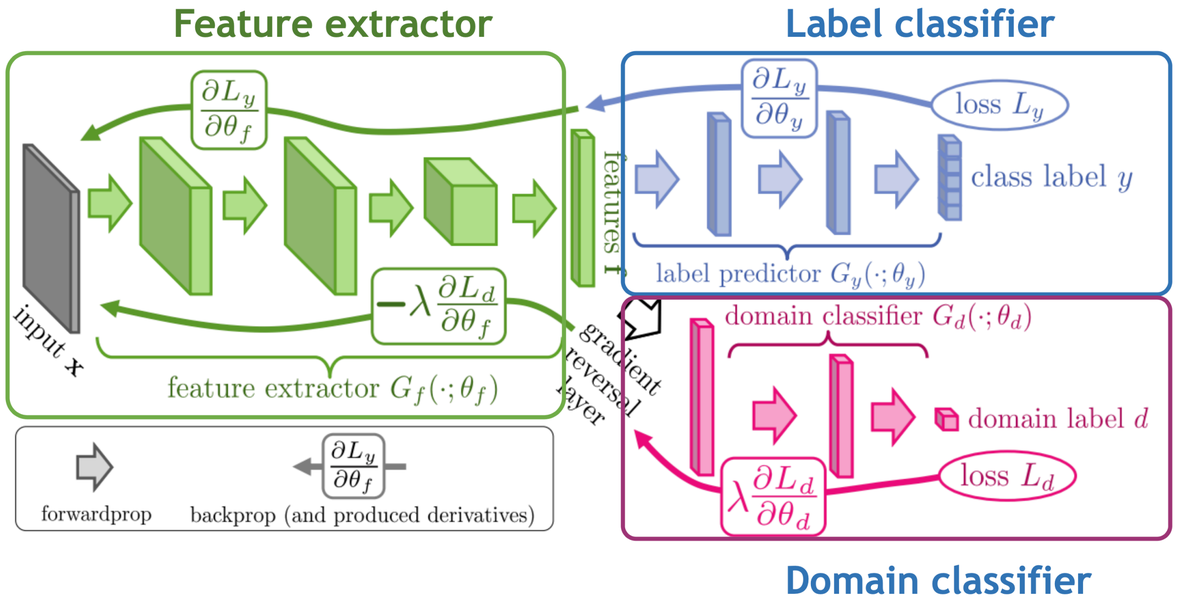
- (2) Source label classifier updates its predictions (**min** loss)

$$\theta = \theta - \eta \nabla_\theta \left\{ E_{(x,y) \sim \mathbb{P}_S} L(h_\theta(\phi_w(x)), y) \right\}$$

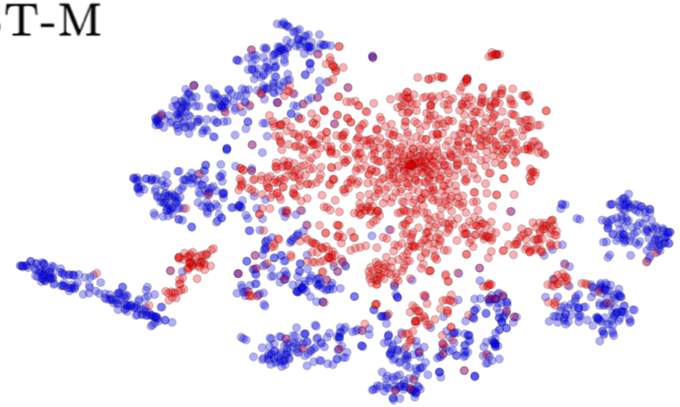
- (3) Representation updates to aid the label classifier (**min** loss), fool the domain classifier (**min** divergence)

$$w = w - \eta \nabla_w \left\{ E_{(x,y) \sim \mathbb{P}_S} L(h_\theta(\phi_w(x)), y) + \lambda \, d(\tilde{\mathbb{P}}_S, \tilde{\mathbb{P}}_T) \right\}$$

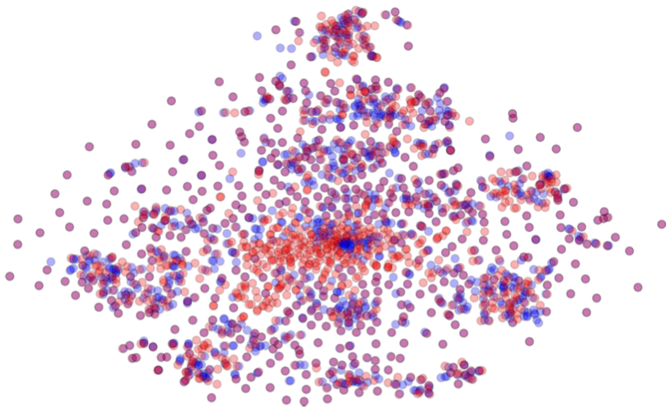
Domain adversarial training



MNIST → MNIST-M: top feature extractor layer



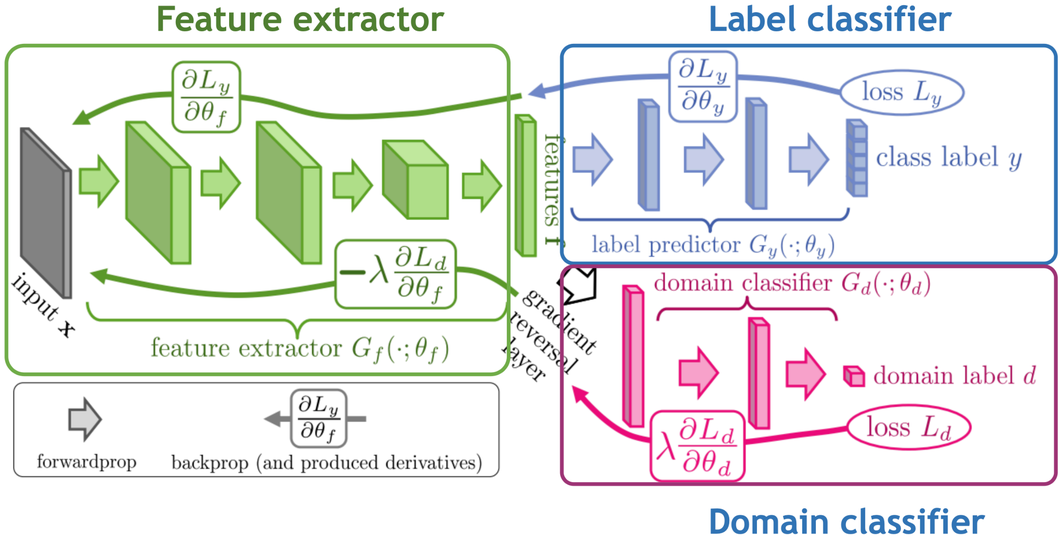
(a) Non-adapted

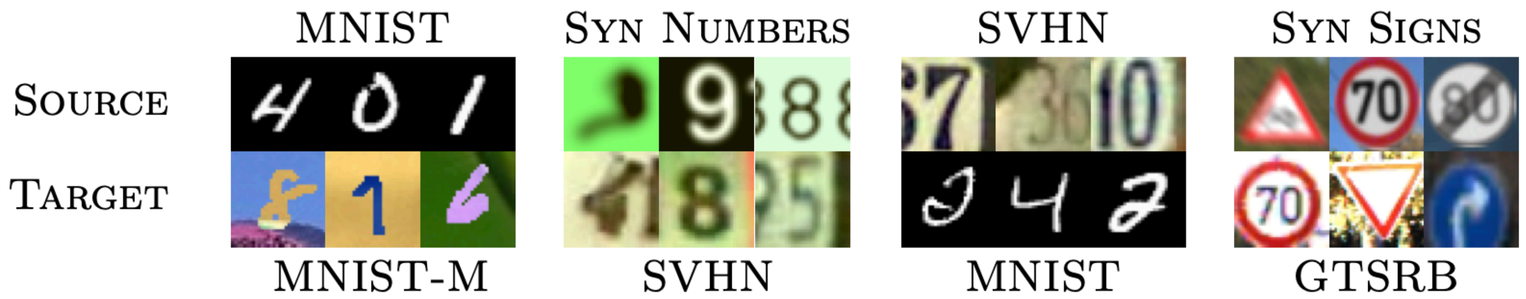


(b) Adapted

[Ganin et al. 2016]

Domain adversarial training





Method	Source / Target	MNIST / MNIST-M	Syn Numbers / SVHN	SVHN / MNIST	Syn Signs / 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. 2016]

Adjusting to three different types of shifts

- Often we can make some assumptions about the type of changes between the domains:

- **Case #1 (label shift only):** the proportions of labels we wish to predict differs from the source to the target domain, the data are otherwise generated the same

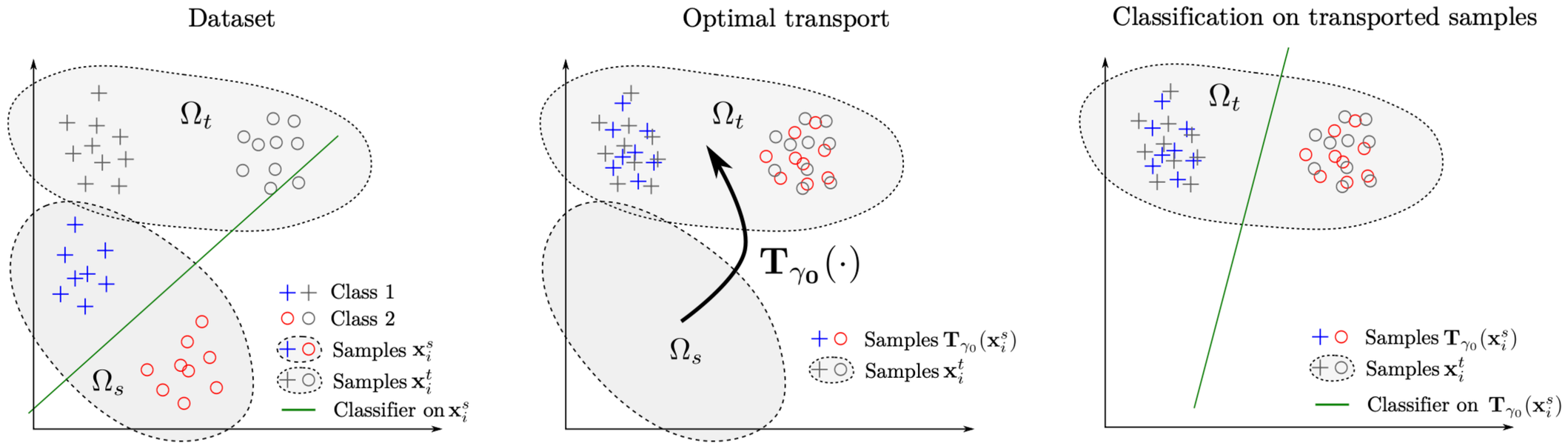
- **Case #2 (co-variate shift only):** the distribution of co-variates (distribution over x's) differs from the source to the target domain, the data are otherwise generated the same

- **Case #3 (co-variate transformation only):** the target domain co-variates are "evolved" or transformed versions of the source co-variates; relation to labels remains after the transformation

- (these are not mutually separable, and the list is not complete)

Co-variate transformations: optimal transport

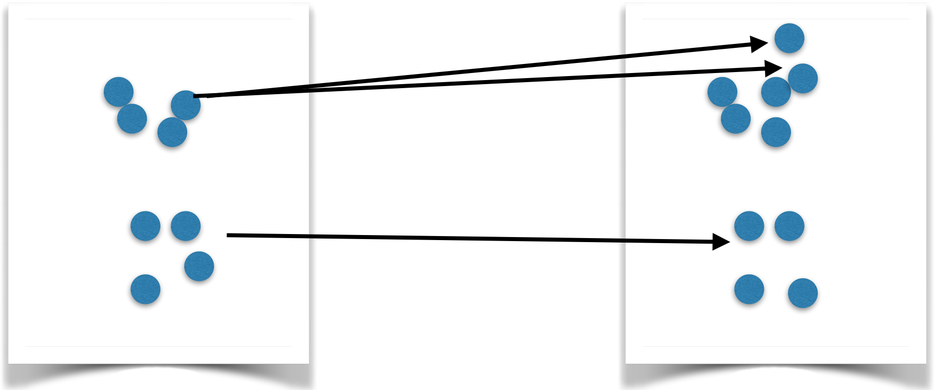
- A different way to think about our problem is to find a way to transport source examples to be (as) target examples, then train and use the predictor there



[figure from Courty et al. 2016]

Discrete optimal transport

each x_i^s
has weight $1/n_s$

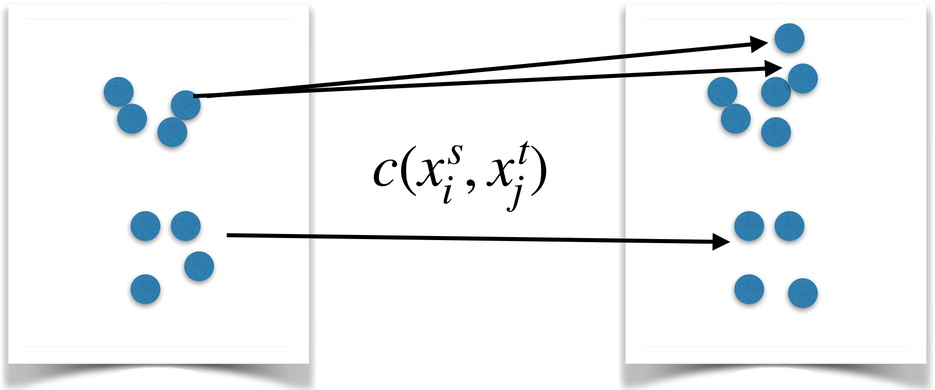


each x_j^t
has weight $1/n_t$

- We wish to minimize the cost of mapping source points (discrete distribution) to target points (discrete distribution), i.e., assume a minimal transformation

Discrete optimal transport

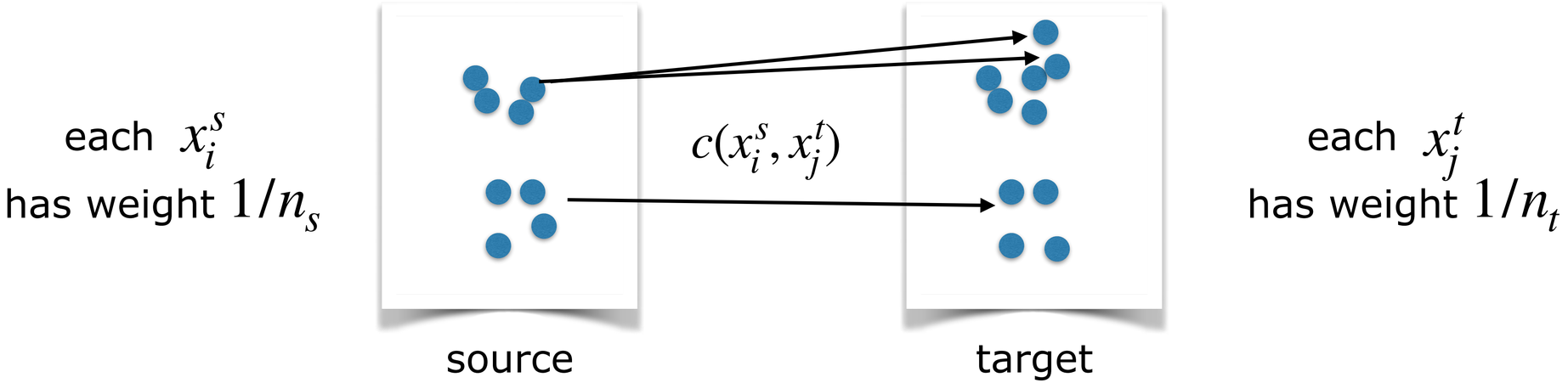
each x_i^s
has weight $1/n_s$



each x_j^t
has weight $1/n_t$

- We wish to minimize the cost of mapping source points (discrete distribution) to target points (discrete distribution), i.e., assume a minimal transformation
- The cost of mapping x_i^s to x_j^t is $c(x_i^s, x_j^t) = \|x_i^s - x_j^t\|^2$ (for example)

Discrete optimal transport

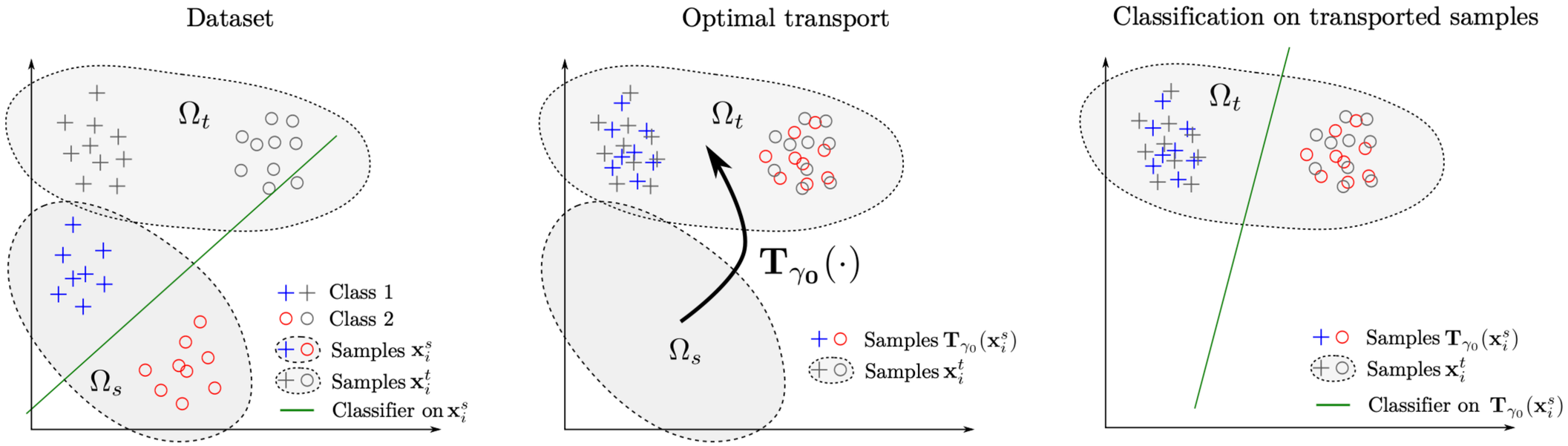


- We wish to minimize the cost of mapping source points (discrete distribution) to target points (discrete distribution), i.e., assume a minimal transformation
- The cost of mapping x_i^s to x_j^t is $c(x_i^s, x_j^t) = \|x_i^s - x_j^t\|^2$ (for example)
- We estimate a non-negative coupling (transport map) $\gamma(i,j)$ that minimizes the transport cost (Wasserstein distance) subject to marginal constraints

$$\min \sum_{i,j} \gamma(i,j)\|x_i^s - x_j^t\|^2 \quad \text{s.t} \quad \sum_j \gamma(i,j) = 1/n_s, \quad \sum_i \gamma(i,j) = 1/n_t$$

Domain adaptation: optimal transport

- We can now either transport source labels to target examples (fractionally, using the transport map) or construct new labeled target examples

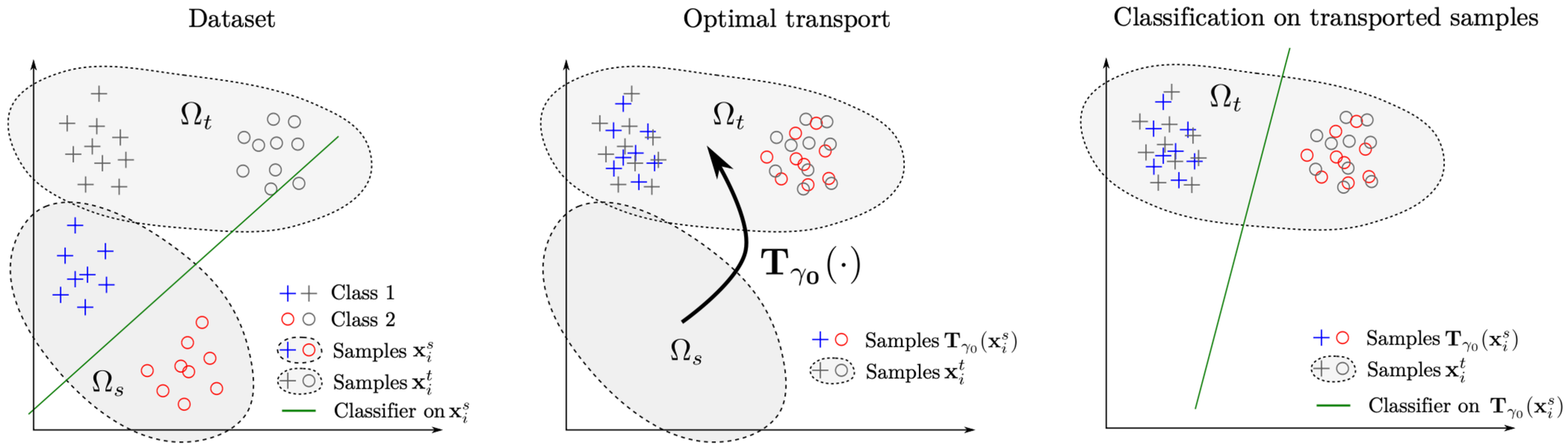


$$\hat{\gamma} = arg\min_{\gamma} \sum_{i,j} \gamma(i,j)\|x_i^s - x_j^t\|^2 \quad \text{s.t} \quad \sum_j \gamma(i,j) = 1/n_s, \quad \sum_i \gamma(i,j) = 1/n_t$$

[Courty et al. 2016]

Domain adaptation: optimal transport

- We can now either transport source labels to target examples (fractionally, using the transport map) or construct new labeled target examples



$$\hat{\gamma} = arg\min_{\gamma} \sum_{i,j} \gamma(i,j)\|x_i^s - x_j^t\|^2 \quad \text{s.t} \quad \sum_j \gamma(i,j) = 1/n_s, \quad \sum_i \gamma(i,j) = 1/n_t$$

$$\hat{x}_i^s = arg\min_{x} \sum_j \hat{\gamma}(i,j)\|x - x_j^t\|^2 \quad \text{(Barycenter reconstruction)}$$

[Courty et al. 2016]

Summary

- Supervised domain adaptation
  - we have lots of labeled examples for the source task, a few labeled examples for the target task
  - simple solution: pre-training on the source, training the small additional part for the new target task (+fine-tuning)
- Unsupervised domain adaptation
  - we have labeled source task examples, only unlabeled target examples
  - we can adjust for label shift, co-variate shift
  - we can also try to “align” the target task to look like the source in the feature space
  - adversarial domain alignment estimates a feature mapping that a) enables source examples to be classified well, b) source and target examples look alike, are distributionally matched
- Domain generalization (cf. out of distribution generalization)
  - in this case we have multiple source domains but the target domain is unknown, might differ similarly from the source domains as the source domains differ from each other

Additional references

- [1] Masashi Sugiyama, Matthias Krauledat, Klaus-Robert Müller, “Covariate Shift Adaptation by Importance Weighted Cross Validation”, Journal of Machine Learning Research (JMLR), 8:985–1005, 2007.
- [2] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. “Domain-adversarial training of neural networks”. Journal of Machine Learning Research (JMLR), 17:2096–2030, 2016
- [3] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, “Optimal transport for domain adaptation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 9, pp. 1853–1865, 2016. https://arxiv.org/abs/1507.00504
- [4] Gulrajani et al., “In search of lost domain generalization”, 2021 https://arxiv.org/pdf/2007.01434.pdf
- [5] G. Peyre and M. Cuturi, “Computational Optimal Transport”, 2018, https://arxiv.org/abs/1803.00567