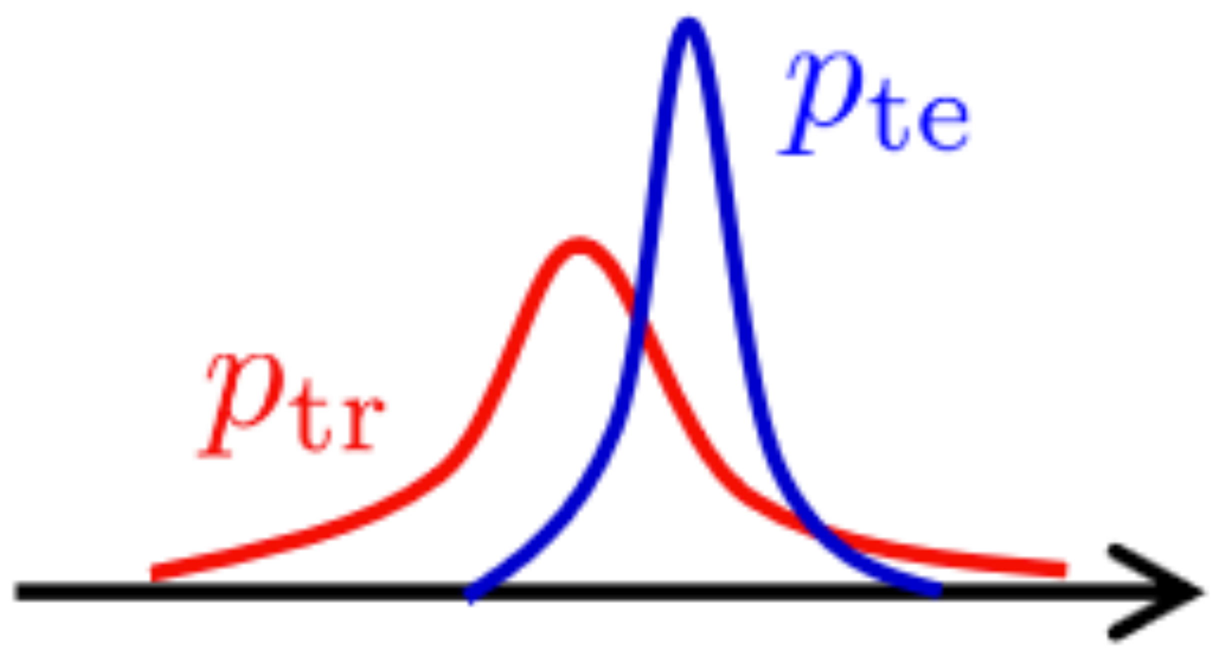
Introduction to Domain Adaptation

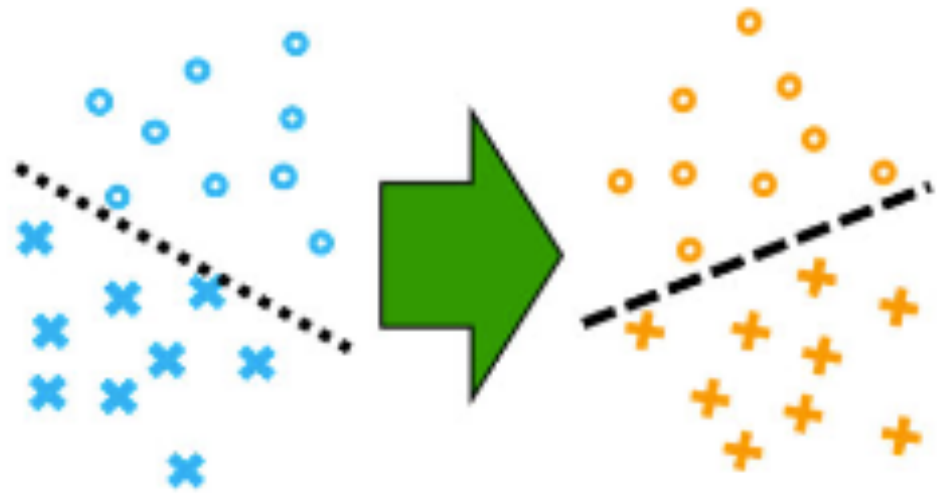
Okan Koc,
RIKEN AIP,
June 26, 2024

Lecture at UTokyo: Special Topics in Mechano-Informatics II

Outline

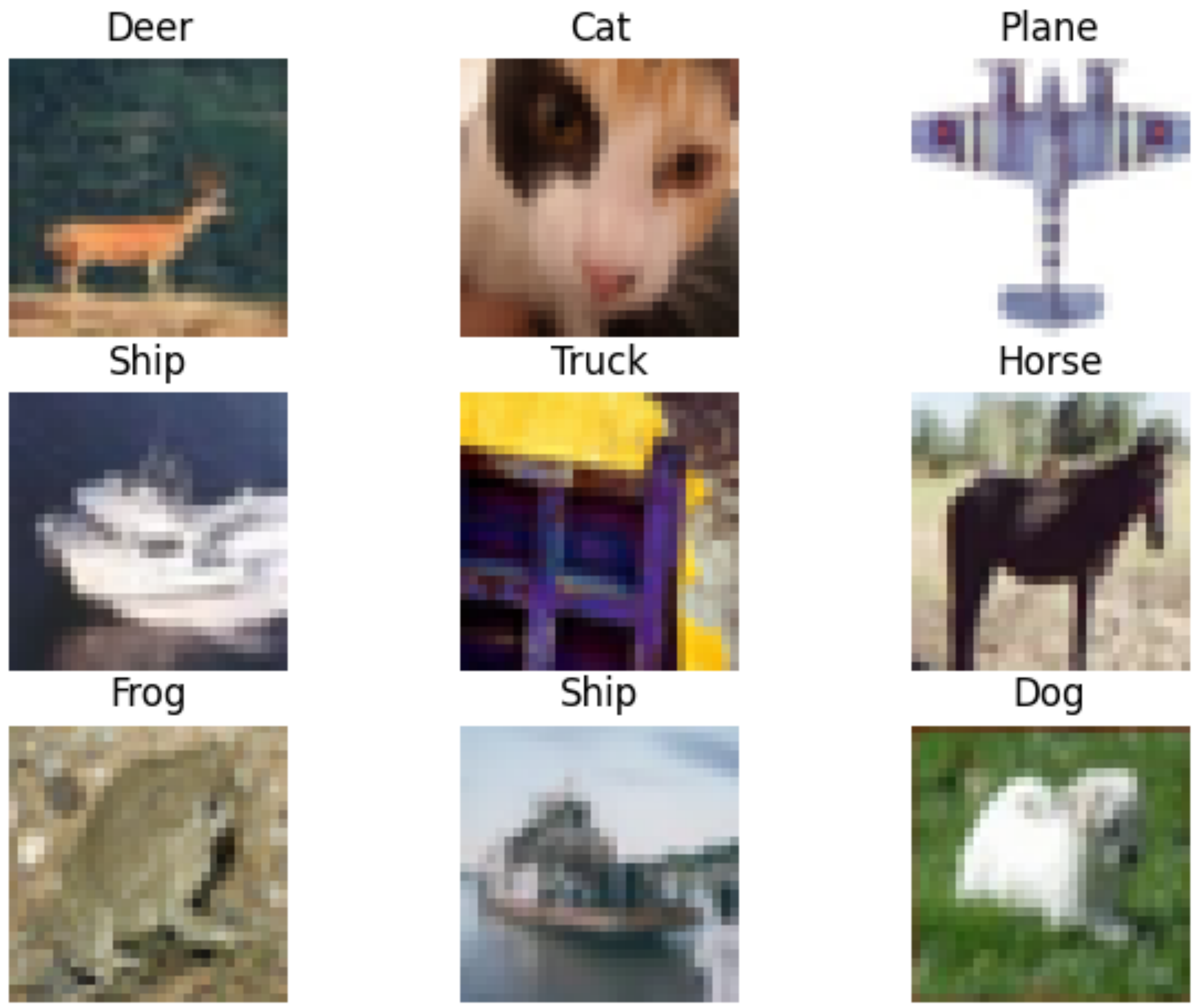
- Empirical Risk Minimization (ERM)
- What is domain adaptation?
  - Analysis of domain adaptation: SBD2010 paper
- Practical methods for domain adaptation
  - For (deep) neural networks: DANN, …
  - Importance weighting (recap)
- Conclusion & Homework





Empirical Risk Minimization

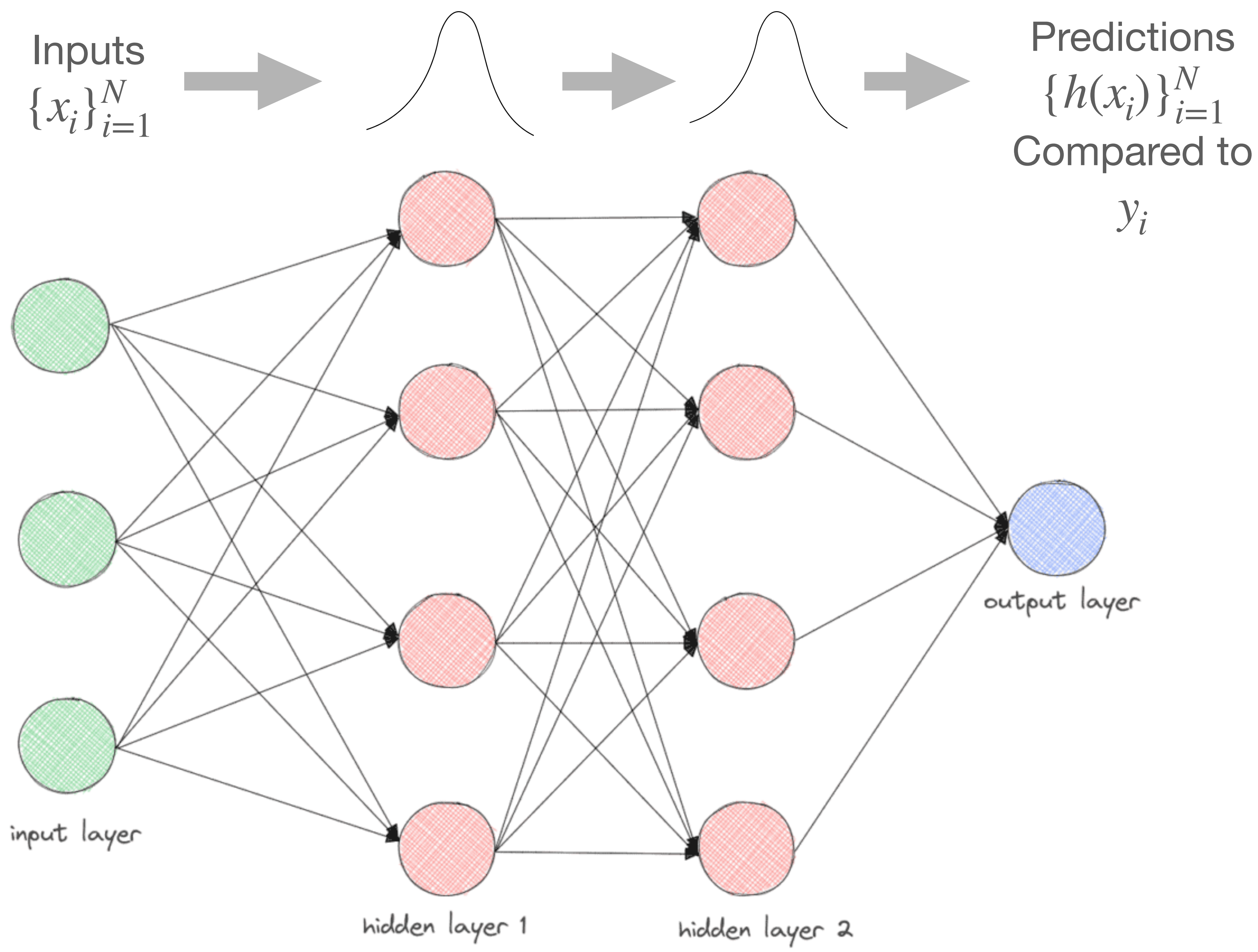
- Input and label pairs $\{x_i, y_i\}_{i=1}^{N}$
  - Ex: Given cat and dog images $x \in \mathbb{R}^{3x32x32}$ learn to distinguish between cats ($y = 4$) and dogs ($y = 6$)
  - using one-hot encoding:
    $y = [0001000000] \in \mathbb{R}^{10}$ or
    $y = [0000010000]$



CIFAR-10 Examples

Empirical Risk Minimization

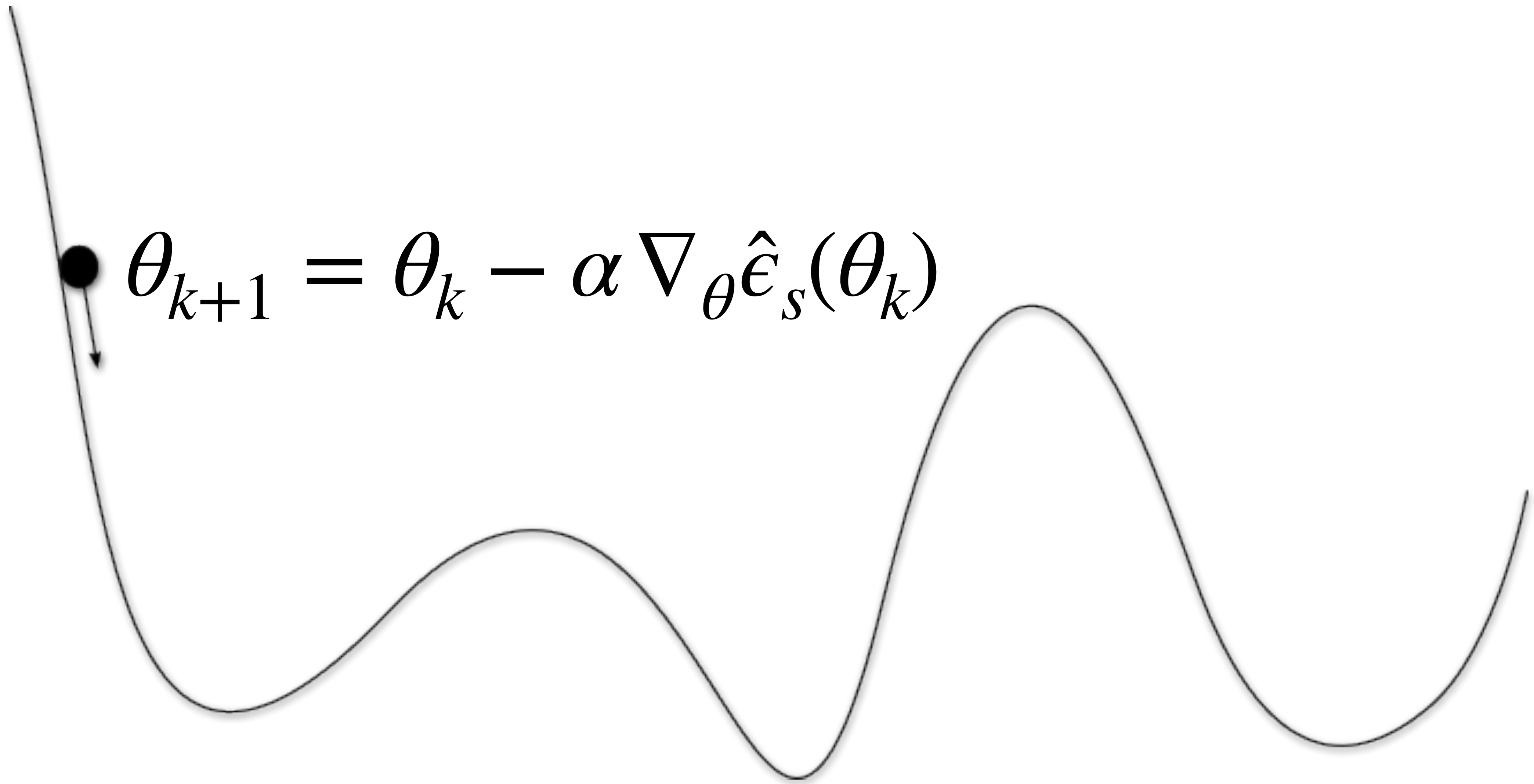
- Hypothesis (model) class $h(x;\theta)$
  - Ex: one hidden-layer neural network $h(x;\theta) = \text{softmax}(W_2\sigma(W_1 x + b_1) + b_2)$
  - Softmax operator: $\text{softmax}(x_i)_j = \frac{e^{x_{ij}}}{\sum_j e^{x_{ij}}}$
- Loss function $l(\hat{y}, y)$,
  - Ex: cross-entropy $-y^T \log \hat{y}$



https://www.baeldung.com/cs/hidden-layers-neural-network

Empirical Risk Minimization

- Minimize empirical risk $\hat{\epsilon}_s(h) = \sum_{i=1}^{N} l(h(x_i;\theta), y_i)$
- Using a practical optimizer,
  - Ex: using SGD, $\theta_{k+1} = \theta_k - \alpha \sum_{i=1}^{B} \nabla_\theta l(h(x_i;\theta_k), y_i)$
  - B is the batch size, α is the learning rate
- And hope that it generalizes well at test time
  - Expected risk $\epsilon_s(h) = \mathbb{E}_{x,y \sim p_s}[l(h(x;\theta), y)]$ should be low! 🙏



Generalization bounds

- Bound the difference using some notion of 'complexity' of hypothesis class
- Given Vapnik-Chernovesky (VC) dimension $d(\mathscr{H})$
- PAC-bound: for any $\delta \in (0,1)$ with probability at least $1 - \delta$
- $\epsilon_S(h) \leq \hat{\epsilon}_S(h) + \sqrt{\frac{d(\mathscr{H})\log(2N) + \log(2/\delta)}{N}}$
- Many other bounds possible (Rademacher complexity, PAC-Bayes bounds, etc.)
- However, bounds assume samples are drawn i.i.d. from $p_s(x, y)$!

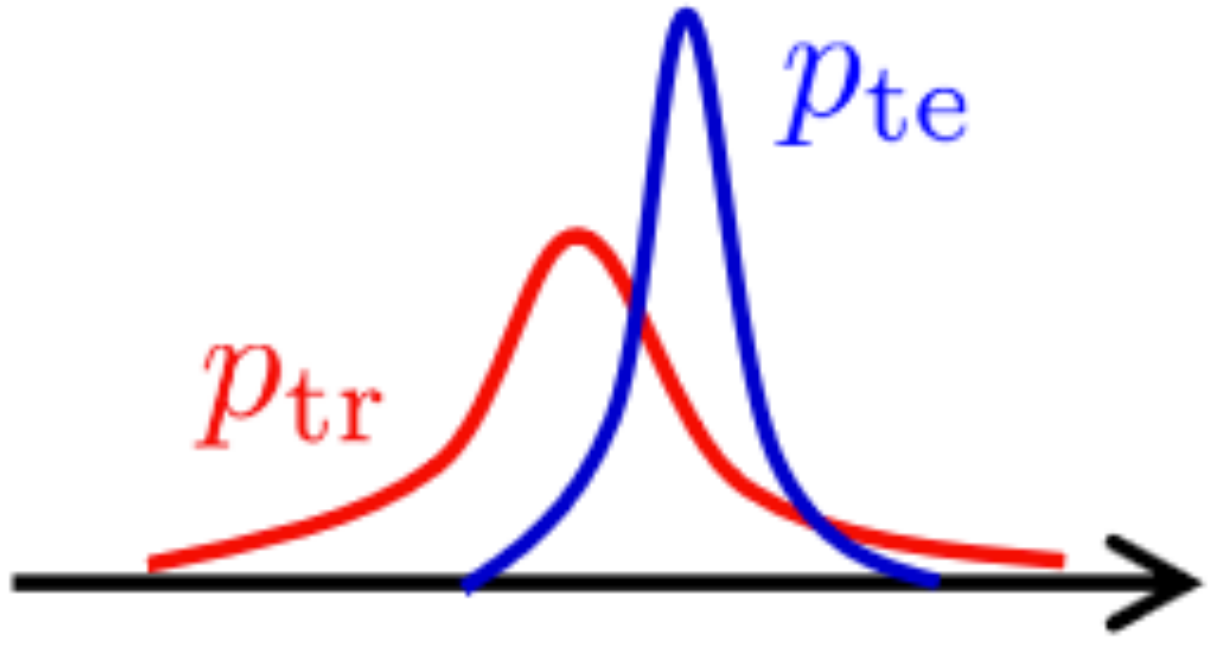
Illustrative example (pytorch)

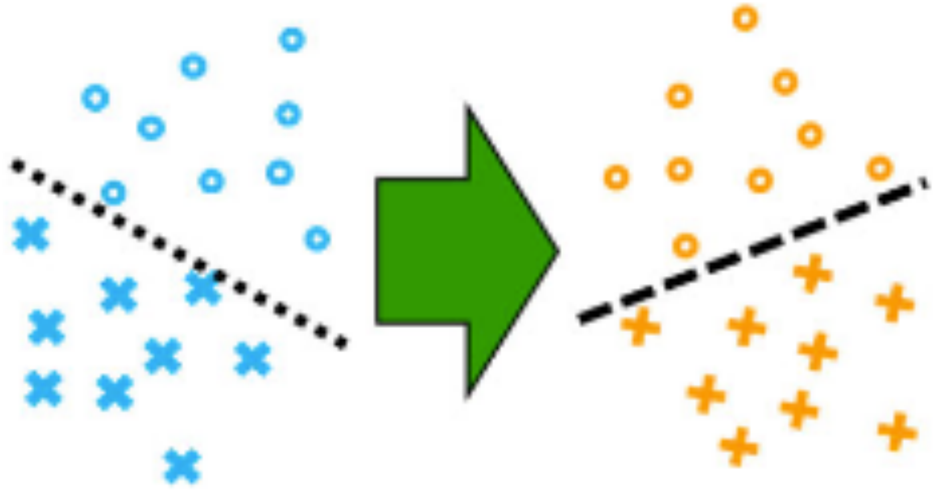
- Sample pytorch code
  - Test and training data are both drawn from same dataset, e.g. CIFAR-10
  - A ‘dataloader’ is used to create mini batches for SGD/Adam
  - Model is a convolutional neural network (CNN)

```
def run_network():
    train_data, test_data = get_data()
    train_dataloader, test_dataloader = get_data_loader(train_data, test_data, batch_size=128)

    model = LeNet().to(DEVICE)
    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

    epochs = 10
    for t in range(epochs):
        train(train_dataloader, model, loss_fn, optimizer)
        test(test_dataloader, model, loss_fn)
```

What is domain adaptation?

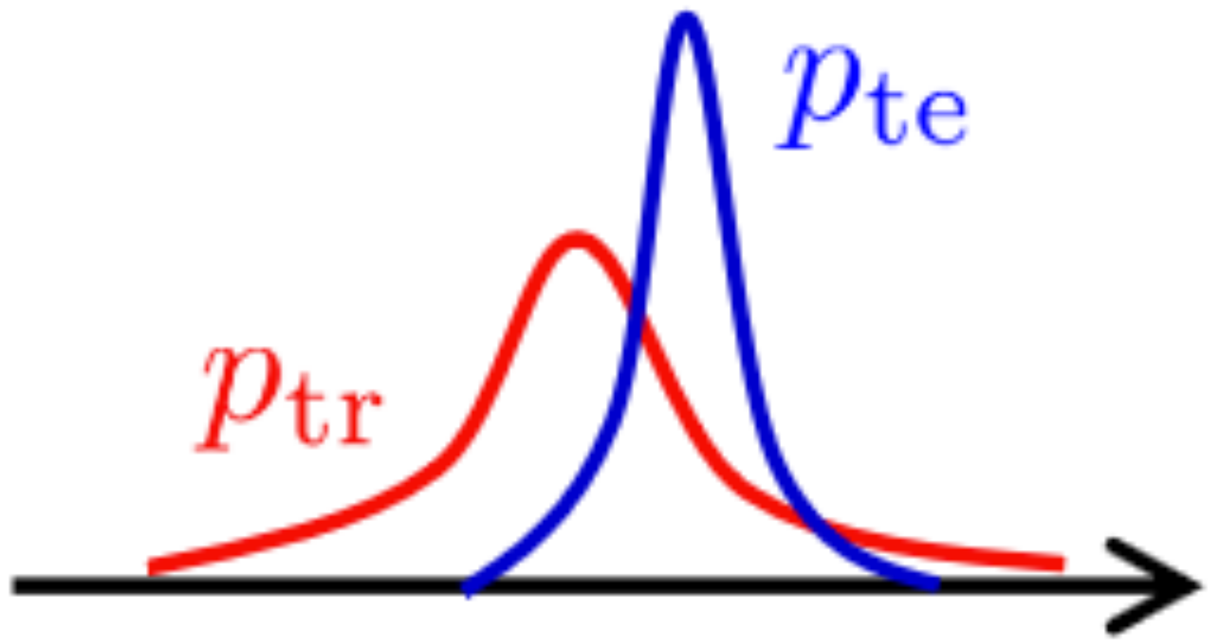
- Part of transfer learning: we want to ‘transfer’ what we learned in one dataset to another
  - But unlike other transfer learning problems, such as multi-task learning, the task is often fixed, e.g. identify digits from 0-9
- Independent and identically distributed assumption violated!
- Source distribution $p_S(x, y)$ changes to target distr. $p_T(x, y)$, called **distribution shift**

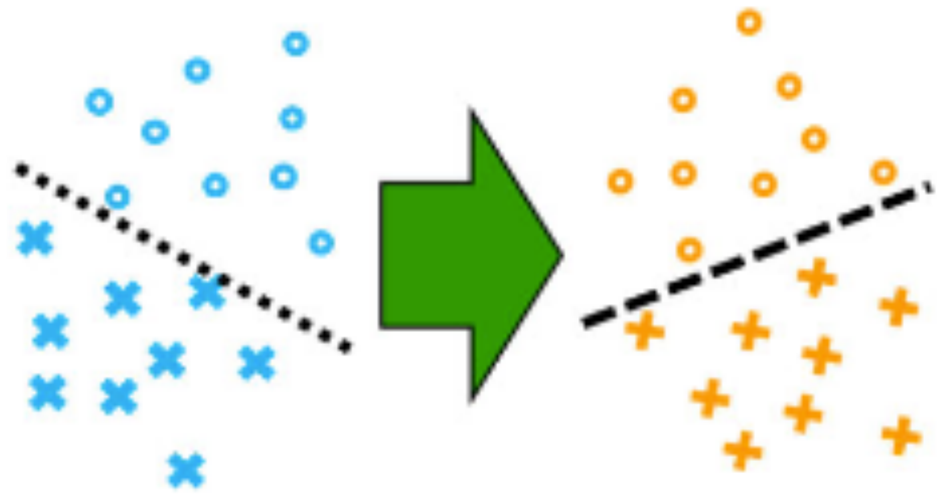




Examples of distribution shift

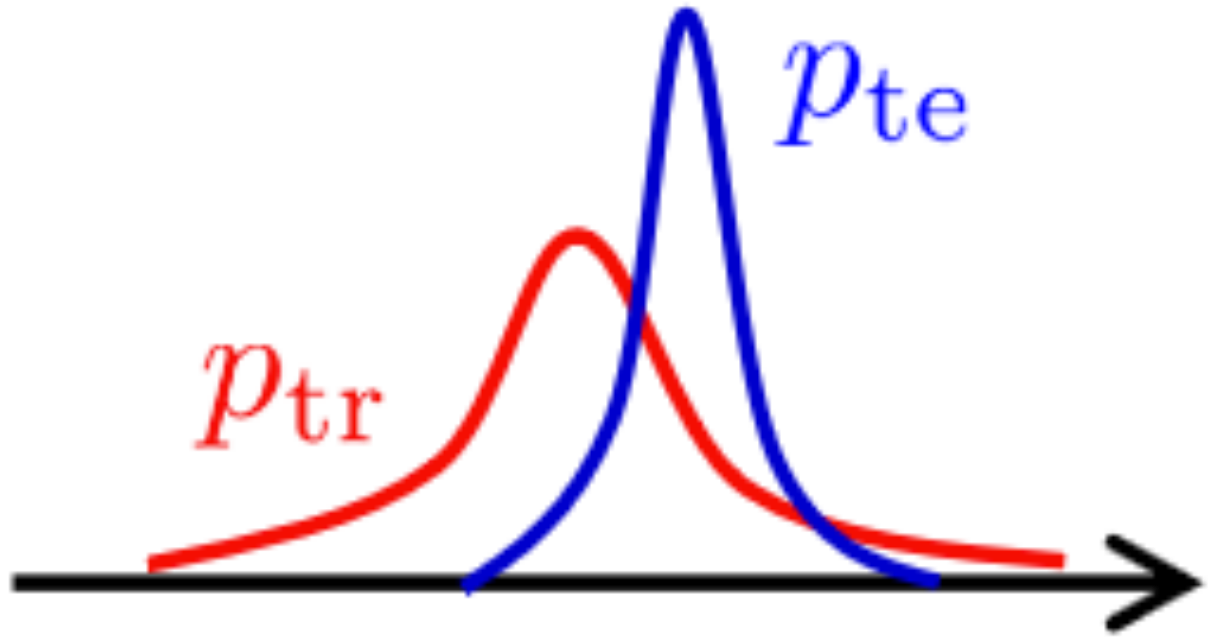
- Detecting objects in the ‘wild’: lighting and background changes,
- Autonomous driving: scene/weather changes,
- Recognizing faces of minorities,
- Detecting cancer cells,
- Many others …

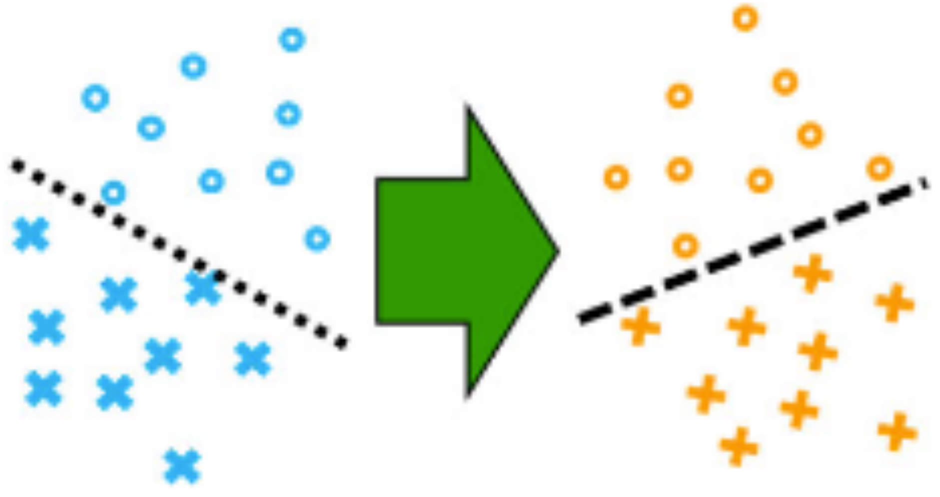




What is domain adaptation?

- Unsupervised case: given (many) samples from $p_S(x, y)$ and (many) unlabeled samples from $p_T(x)$
- Semi-supervised case: additionally given few labelled samples from $p_T(x, y)$
- Covariate shift assumption: the concept does not change in a lot of problems (e.g. digit recognition). Formalize as: $p_T(y \mid x) = p_S(y \mid x)$,
    - so only the marginals $p_S(x)$ or $p_S(y)$ can change





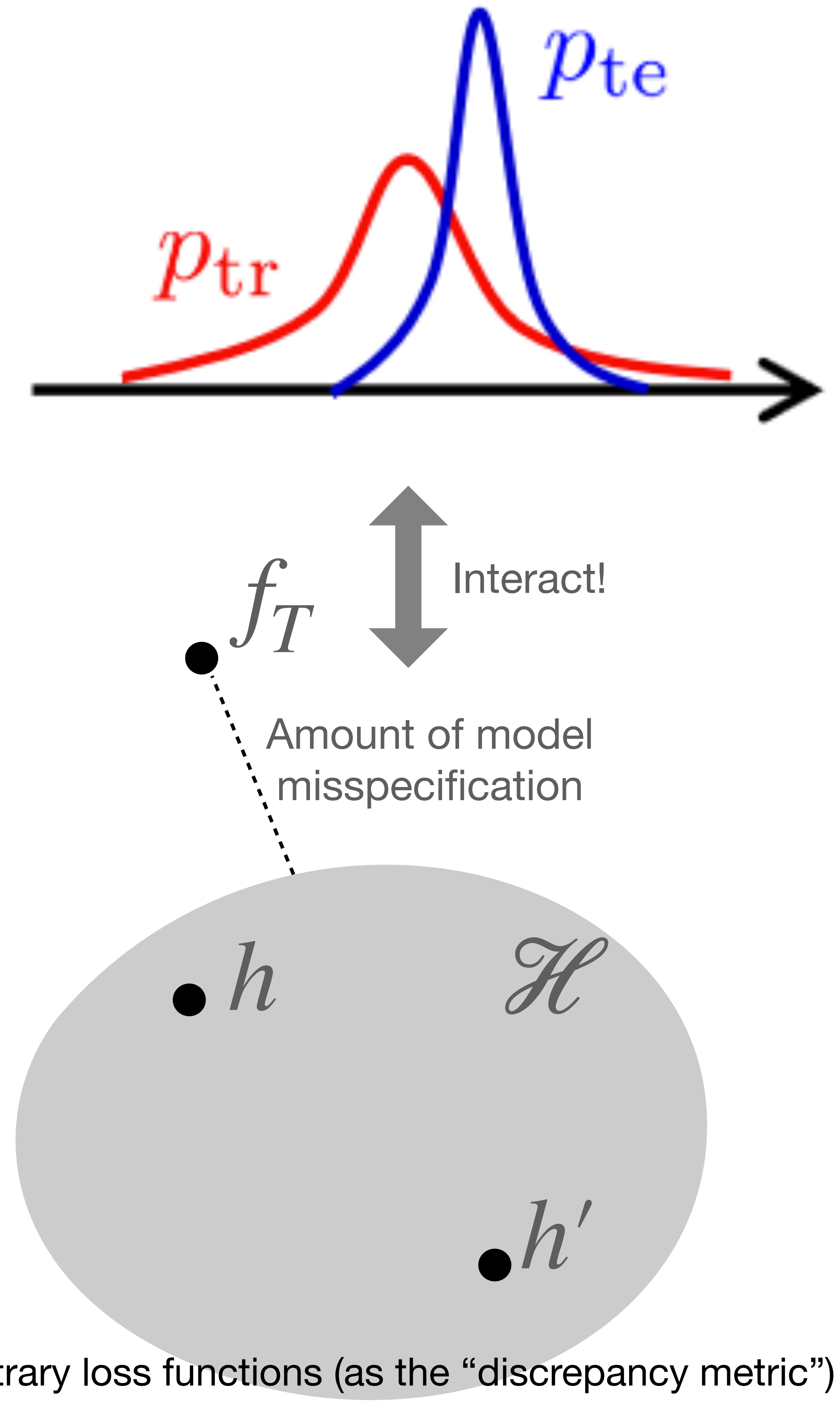
Distances between probability distributions

- Intuitively, if $p_S(x, y)$ and $p_T(x, y)$ are significantly different from each other, there is no hope to transfer any knowledge!

- Hence the 'distance' between the distributions should appear in generalization bounds

- $H\Delta H$-divergence*:
$$d_{H\Delta H}(p_S, p_T) = \sup_{h,h' \in \mathcal{H}} \left| \epsilon_S(h, h') - \epsilon_T(h, h') \right|$$

- Change in notation:
$$\epsilon_S(h, f_S) := \mathbb{E}_{x \sim p(x), y = f_S(x)}[l(h(x), f_S(x))]$$



*Original formulation in [SBD2010] considered the 0-1 loss and binary classification which is generalized in [Mansour2009] to arbitrary loss functions (as the "discrepancy metric")

Distances between probability distributions

- A distance (a.k.a. metric) between x and y, d(x, y) should obey
  - Nonnegativity: $d(x, y) \geq 0$
  - Identity of indiscernibles: $d(x, y) = 0 \leftrightarrow x = y$
  - Symmetry: $d(x, y) = d(y, x)$
  - Triangle inequality: $d(x, y) \leq d(x, z) + d(z, y)$

‘Divergences’ violate the triangle inequality

Distances between probability distributions

- Generalize from vector space to probability distributions p(x) and q(x)
  - L1 distance: $d_1(p, q) = \int |p(x) - q(x)| dx$
  - KL divergence: $KL(p || q) = \int p(x) \log \frac{p(x)}{q(x)} dx$
  - Many other distances/divergences possible, IPMs, Renyi divergence, Bregman divergences etc.

$$IPM(p, q; \mathscr{H}) = \sup_{h \in \mathscr{H}} \mathbb{E}_{x \sim p}[h(x)] - \mathbb{E}_{x \sim q}[h(x)] = \sup_{h \in \mathscr{H}} \int h(x) \big(p(x) - q(x)\big) dx$$

Distances between probability distributions

- $d_1(p,q) = \int |p(x) - q(x)|\, dx$ can be very big, difficult to estimate from samples!
- $KL(p\,||\,q) = \int p(x) \log \frac{p(x)}{q(x)} dx$ requires same support, otherwise infinite!
- KL is not a metric (violates triangle inequality)!
- Ex: for univariate Gaussians $p_S(x) = \mathcal{N}(\mu_S, \sigma_S^2)$ and $p_T(x) = \mathcal{N}(\mu_T, \sigma_T^2)$

$$KL(p_S \| p_T) = \frac{1}{2}\left[\log\left(\frac{\sigma_T^2}{\sigma_S^2}\right) + \frac{\sigma_S^2}{\sigma_T^2} + \frac{(\mu_S - \mu_T)^2}{\sigma_T^2} - 1\right]$$

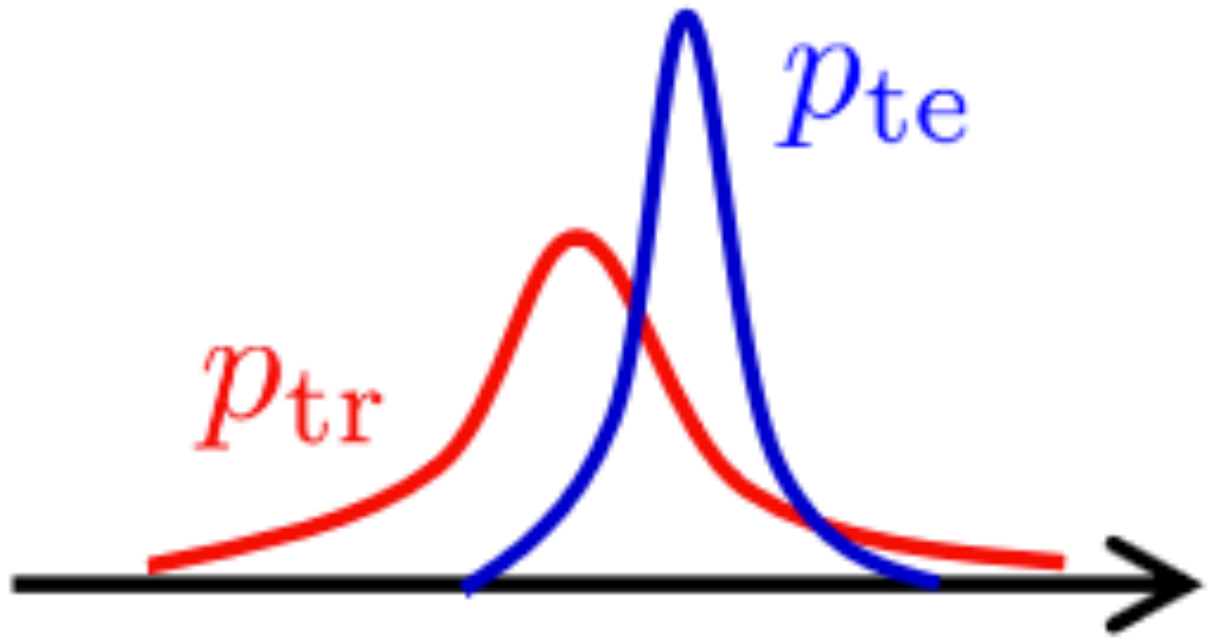
Simple example for $H\Delta H$ divergence

- $H\Delta H$-divergence:
  $$d_{H\Delta H}(S,T) = \sup_{h,h' \in \mathscr{H}} \left| \epsilon_S(h,h') - \epsilon_T(h,h') \right|$$

- Let's consider one of the simplest examples in 1D!

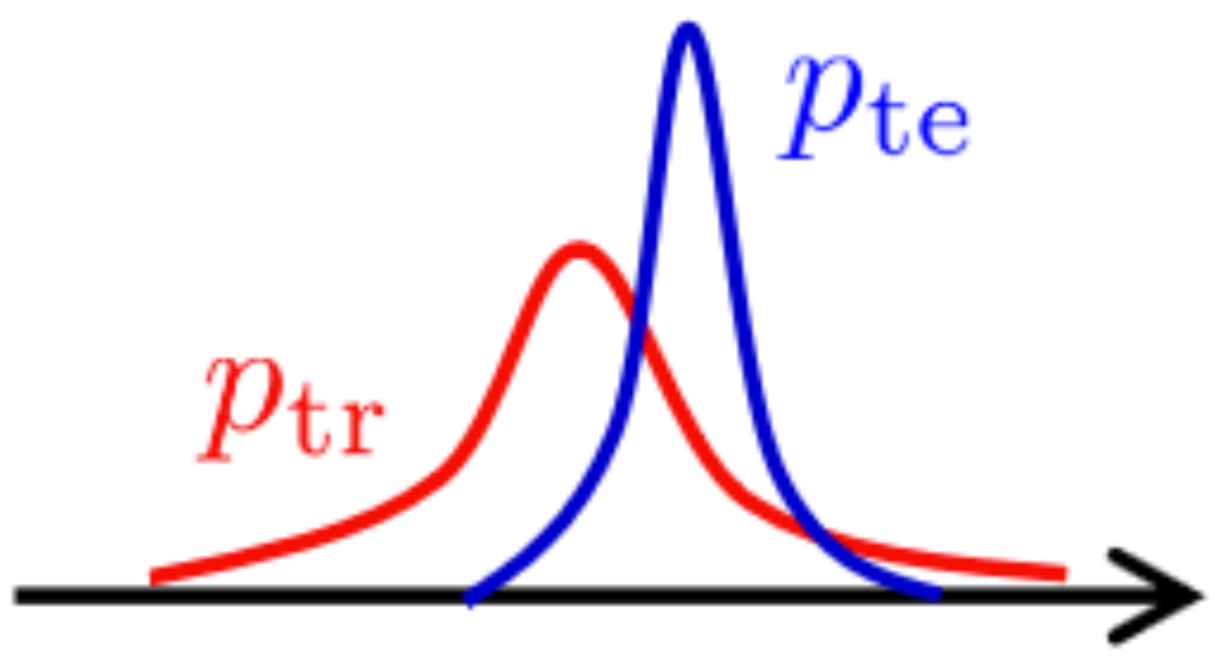
- Consider $p_S(x) = \mathcal{N}(\mu_S, \sigma_S^2)$ shifted to $p_T(x) = \mathcal{N}(\mu_T, \sigma_T^2)$, $p_S(y \mid x) = \mathcal{N}(\theta^* x, \sigma_y^2)$ invariant.

- Using linear regression, i.e. $\epsilon_S(\theta) = \mathbb{E}_S[(y - \theta x)^2]$

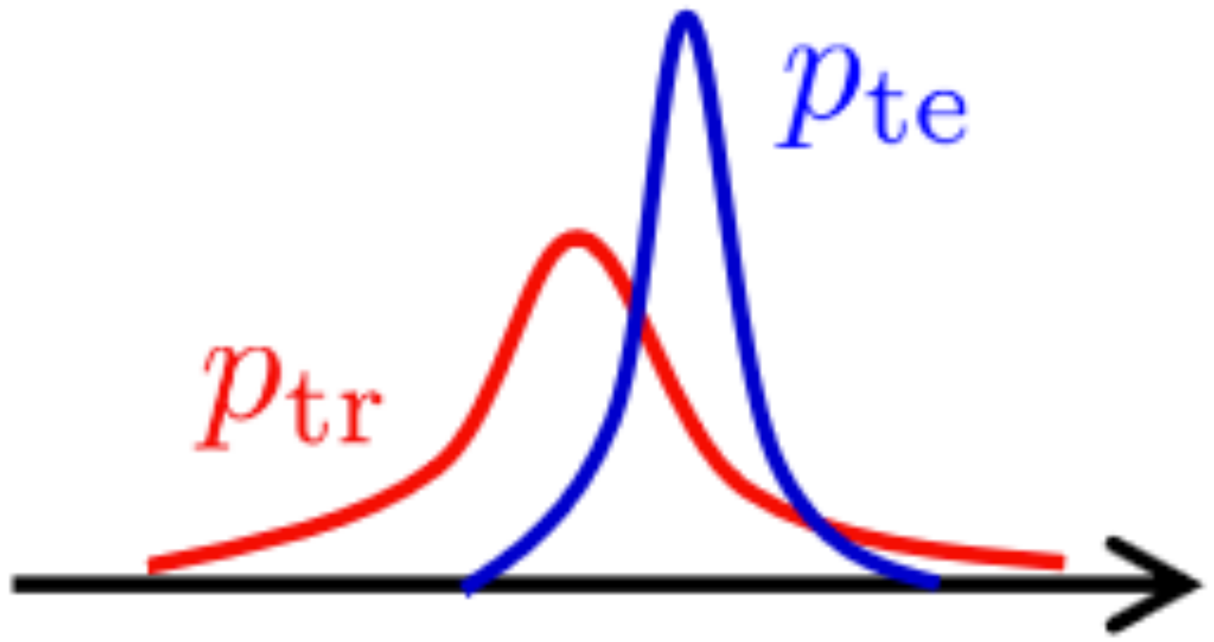


Compute $H\Delta H$ divergence for 1D regression

- Consider $p_S(x) = \mathcal{N}(\mu_S, \sigma_S^2)$ shifted to $p_T(x) = \mathcal{N}(\mu_T, \sigma_T^2)$
- Using squared-loss
- $\epsilon_S(h, h') = \int (\theta - \theta')^2 x^2 p_S(x) dx = (\mu_S^2 + \sigma_S^2)(\theta - \theta')^2$
- Similarly $\epsilon_T(h, h') = (\mu_T^2 + \sigma_T^2)(\theta - \theta')^2$



Compute $H\Delta H$ divergence for 1D regression



- $\epsilon_S(h, h') = (\mu_S^2 + \sigma_S^2)(\theta - \theta')^2$
- $\epsilon_T(h, h') = (\mu_T^2 + \sigma_T^2)(\theta - \theta')^2$
- $d_{H\Delta H}(p_S, p_T) = \sup_{h,h' \in \mathscr{H}} \left| \epsilon_S(h, h') - \epsilon_T(h, h') \right| = \sup_{\theta,\theta' \in (-B,B)} |\mu_T^2 + \sigma_T^2 - \mu_S^2 - \sigma_S^2| (\theta - \theta')^2$
- Bounded parameter space* $\theta \in (-B, B)$
- $d_{H\Delta H}(p_S, p_T) = 4B^2 |\mu_T^2 + \sigma_T^2 - \mu_S^2 - \sigma_S^2|$, small if B is small and shift is small!

*Well specified model if $\theta^* \in (-B, B)$!

Analysis of domain adaptation

- Theorem 1 [SBD2010]. $\epsilon_T(h) \leq \epsilon_S(h) + d_{H\Delta H}(p_S, p_T) + \lambda$

  where $\lambda = \epsilon_S(h^*) + \epsilon_T(h^*)$, $h^* = \arg\min_{h \in \mathscr{H}} \epsilon_S(h) + \epsilon_T(h)$
- Proof: Apply triangle inequality twice to a symmetric loss function

$$
\begin{aligned}
\epsilon_T(h) &\leq \epsilon_T(h^*, f_T) + \epsilon_T(h, h^*) \quad (\Delta\text{-ineq. using } h^*) \\
&\leq \epsilon_T(h^*, f_T) + \epsilon_S(h, h^*) + |\epsilon_T(h, h^*) - \epsilon_S(h, h^*)| \\
&\leq \epsilon_T(h^*) + \epsilon_S(h, h^*) + d_{H\Delta H}(p_S, p_T) \\
&\leq \epsilon_T(h^*, f_T) + \epsilon_S(h, f_S) + \epsilon_S(h^*, f_S) + d_{H\Delta H}(p_S, p_T) \quad (\Delta\text{-ineq. using } f_S \text{ and use sym.}) \\
&= \epsilon_S(h) + d_{H\Delta H}(p_S, p_T) + \lambda
\end{aligned}
$$

Analysis of domain adaptation

- Estimate $d_{H\Delta H}(p_S, p_T)$ from N samples: $\hat{d}_{H\Delta H}(p_S, p_T)$
- The complexity measure of hypothesis class: $d(\mathscr{H})$
- Lemma 1 [SBD2010]. Given N samples from both distr., with prob. at least $1 - \delta$,

$$d_{H\Delta H}(p_S, p_T) \leq \hat{d}_{H\Delta H}(p_S, p_T) + 4\sqrt{\frac{d(\mathscr{H})\log(2N) + \log(2/\delta)}{N}}$$

- Proof: Bound using VC-dimension, see Kifer et al. (2004) for details.

Analysis of domain adaptation

- Theorem 2 [SBD2010].

$$\epsilon_T(h(\theta)) \le \epsilon_S(h(\theta)) + \hat{d}_{H\Delta H}(S,T) + 4\sqrt{\frac{2d\log(2N) + \log(2/\delta)}{N}} + \lambda$$

- Proof: Bound the empirical H-divergence using Lemma 1
- Extend bound to $\hat{\epsilon}_S(h(\theta))$ by another $\tilde{\mathcal{O}}\left(1/\sqrt{N}\right)$ term from i.i.d generalization bound
- $\epsilon_T(h(\theta)) \le \hat{\epsilon}_S(h(\theta)) + \hat{d}_{H\Delta H}(S,T) + \lambda + c\sqrt{\frac{\log N}{N}}$ for some constant c

Revisiting our example

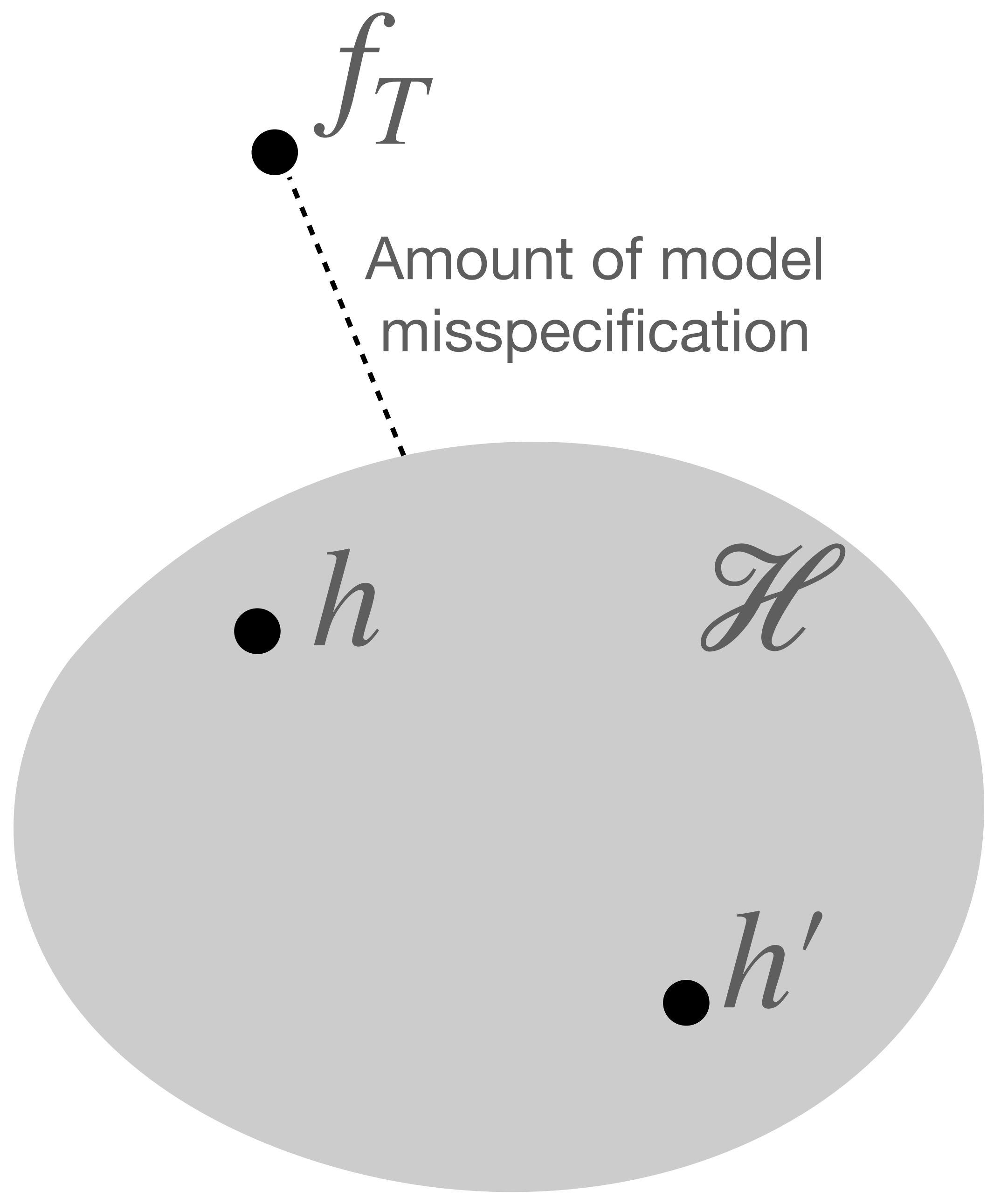
- Consider $p_S(x) = \mathcal{N}(\mu_S, \sigma_S^2)$ shifted to $p_T(x) = \mathcal{N}(\mu_T, \sigma_T^2)$,
- $p_S(y|x) = p_T(y|x) = \mathcal{N}(\theta^* x, \sigma_y^2)$ stays the same.
- Using linear regression (well-specified model!) and squared-loss
- $\epsilon_S(\theta) = \mathbb{E}_S[(y - \theta x)^2] = (\mu_S^2 + \sigma_S^2)(\theta - \theta^*)^2 + \sigma_y^2$
- $\epsilon_T(\theta) = \mathbb{E}_T[(y - \theta x)^2] = (\mu_T^2 + \sigma_T^2)(\theta - \theta^*)^2 + \sigma_y^2$
- $\hat{d}_{H\Delta H}(p_S, p_T) = 4B^2|\hat{\mu}_T^2 + \hat{\sigma}_T^2 - \hat{\mu}_S^2 - \hat{\sigma}_S^2|$
- $\lambda = \min_{\theta \in [-B,B]} \epsilon_T(\theta) + \epsilon_S(\theta) = 2\sigma_y^2$

$$\epsilon_T(\hat{\theta}) \leq \hat{\epsilon}_S(\hat{\theta}) + 4B^2 a + b + \frac{c \log N}{\sqrt{N}}$$

$\Rightarrow$ Choose B wisely!

Revisiting our example

- Using linear regression (well-specified model!) and squared-loss
- $\epsilon_T(\hat{\theta}) \leq \hat{\epsilon}_S(\hat{\theta}) + 4B^2 a + b + \frac{c \log N}{\sqrt{N}}$
- Conclusion was: choose B wisely!
- Models are almost always (at least a little) *misspecified*!
- However $d_{H\Delta H}(p_S, p_T)$ divergence is agnostic to model-misspecification!
- If model is misspecified, ERM may not find a small $\hat{\epsilon}_S(\hat{\theta})$! Likewise λ may not be small!

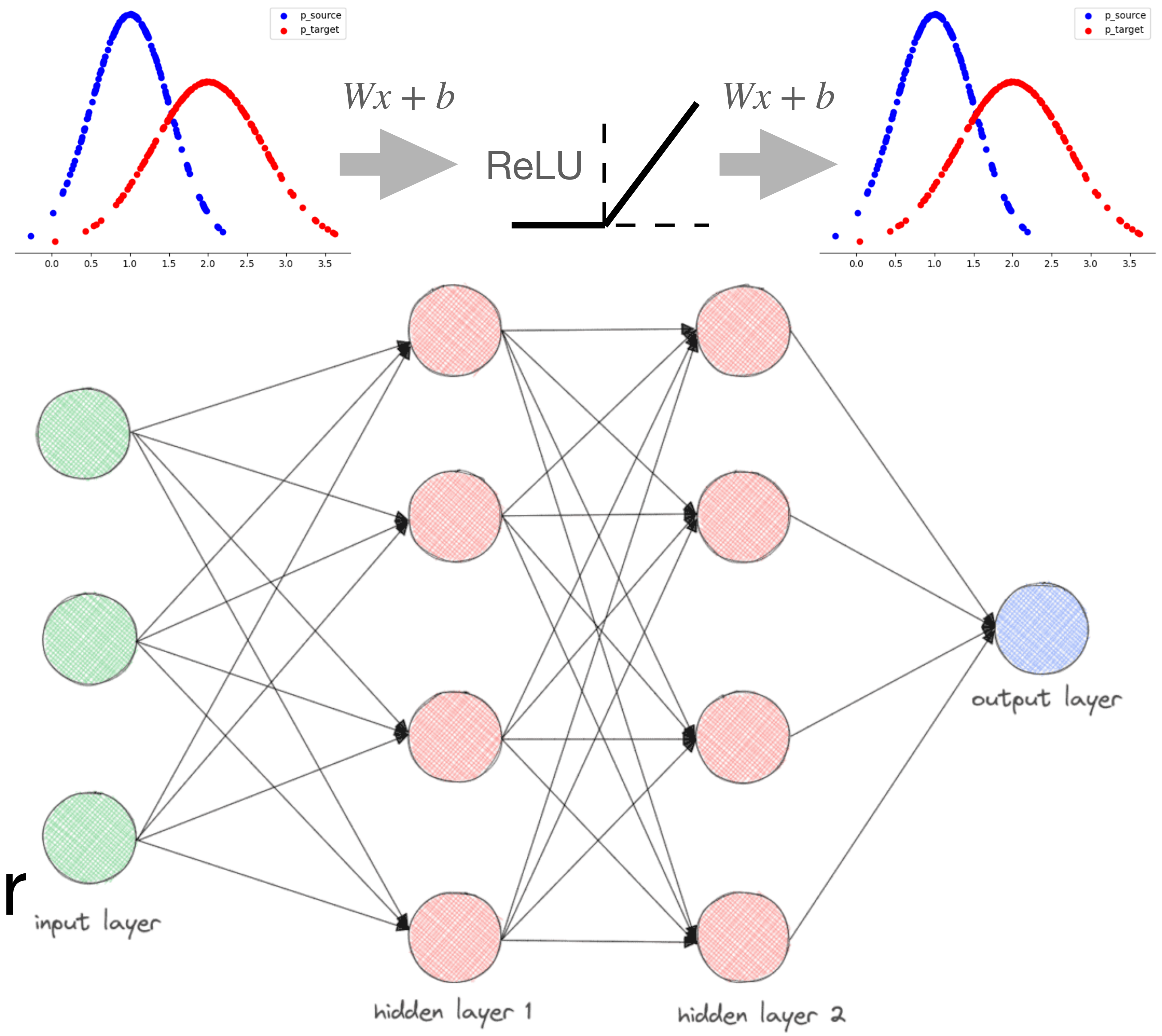


Practical algorithms for domain adaptation

- Moment matching
- Practical methods based on upper-bound of target risk
  - Adversarial approaches
  - Regularization based methods (e.g. Wasserstein-distance)
- Alternative: Importance weighting

Moment matching methods

- Match-moments of source and target distributions in the feature space
    - [CORAL2016] matches covariances (after whitening)
    - [BN2021] Test-time batch-normalization, …
- These methods try to find representations such that the first two moments $\mu_{S,t}, \Sigma_{S,t}$ in each layer t are as similar as possible to $\mu_{T,t}, \Sigma_{T,t}$



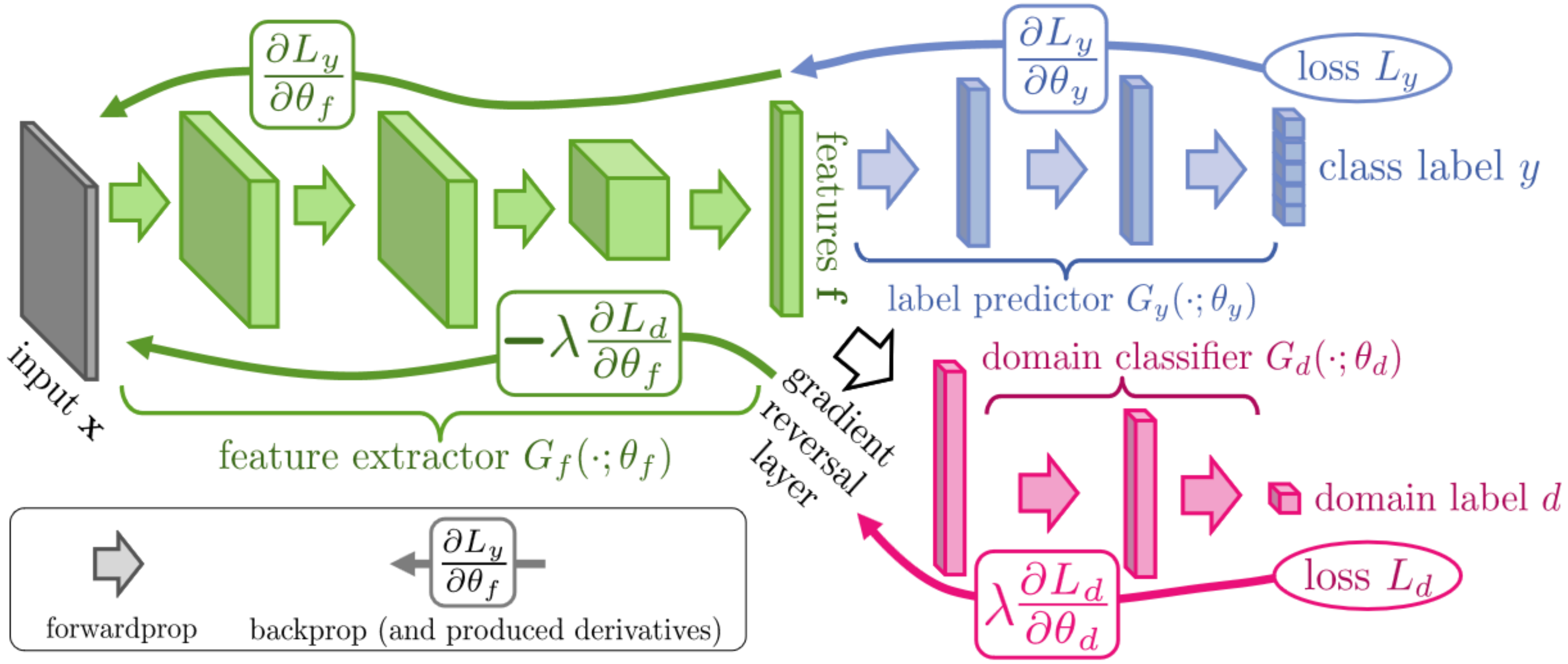
$$\mu_{S,t}(\theta) \approx \mu_{T,t}(\theta)$$
$$\Sigma_{S,t}(\theta) \approx \Sigma_{S,t}(\theta)$$

Practical methods based on upper-bound

- $H\Delta H$-divergence does not immediately yield a *computable* upper bound!
  - Ex: In our 1D-regression example $\epsilon_T(\hat{\theta}) \leq \hat{\epsilon}_S(\hat{\theta}) + 4B^2a + b + \frac{c \log N}{\sqrt{N}}$, a might be estimated from data but b is not known!
  - The error λ of "best possible hypothesis in both environments" is not known!
  - $H\Delta H$ divergence is not easy to compute, or computable upper bound might be very loose!

An adversarial approach

- Adversarial approach: DANN!
- Implemented as three Neural Networks: one *feature extractor* G_f, one *label predictor* G_y and one *domain classifier* G_d
- Inspired by $H\Delta H$-divergence but ignoring the λ term!



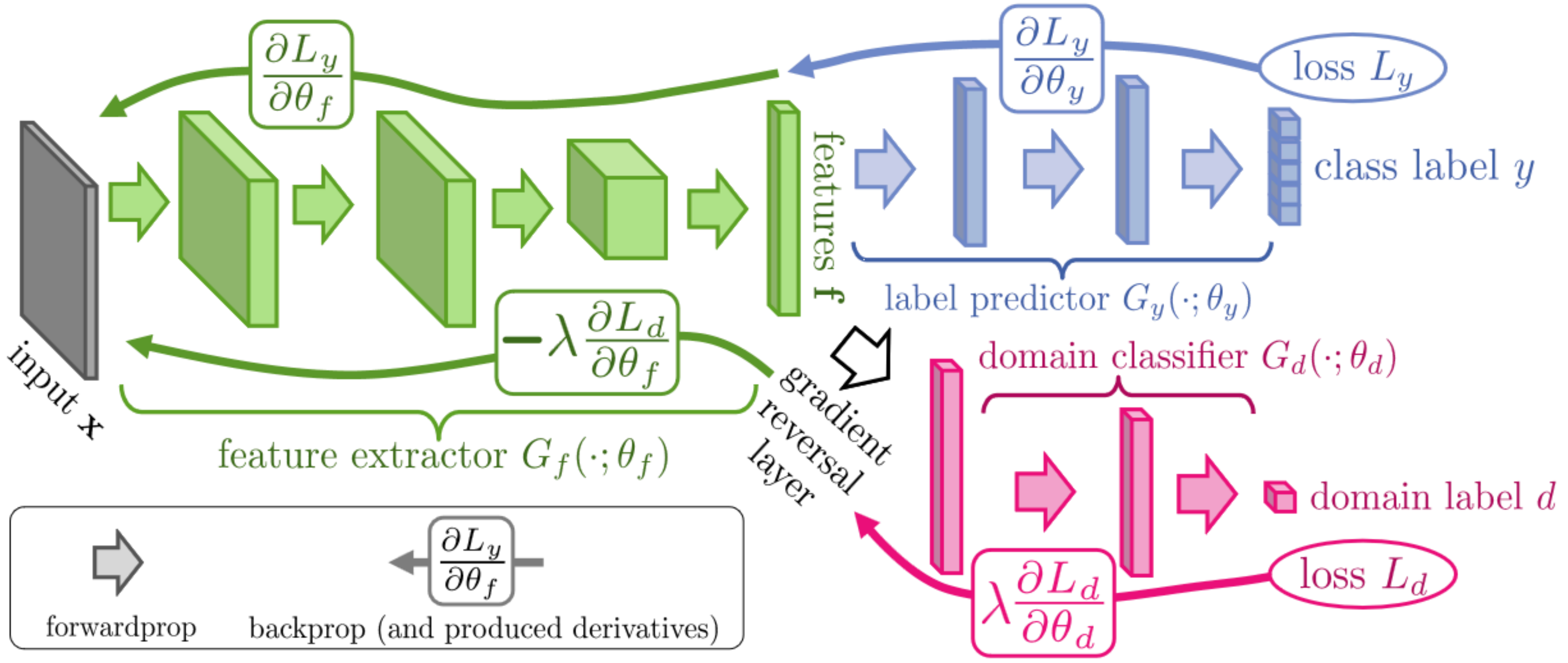
Proposed architecture of the Domain Adversarial Neural Networks (DANN) algorithm

A practical algorithm for domain adaptation

- Main argument: learn a feature extractor which does not allow the domain classifier to distinguish (i.e. classify successfully!) between the *source* and the *target* inputs!

- Can be cast as a min-max optimization with a saddle point as optimum!

- For the binary case, minimize cross entropy



Proposed architecture of the Domain Adversarial Neural Networks (DANN) algorithm

$$L_d(G_d(G_f(x_i)), d_i) = \sum_{i=1}^{N} - d_i \log G_d(G_f(x_i)) - (1 - d_i)\log\big(1 - G_d(G_f(x_i))\big)$$

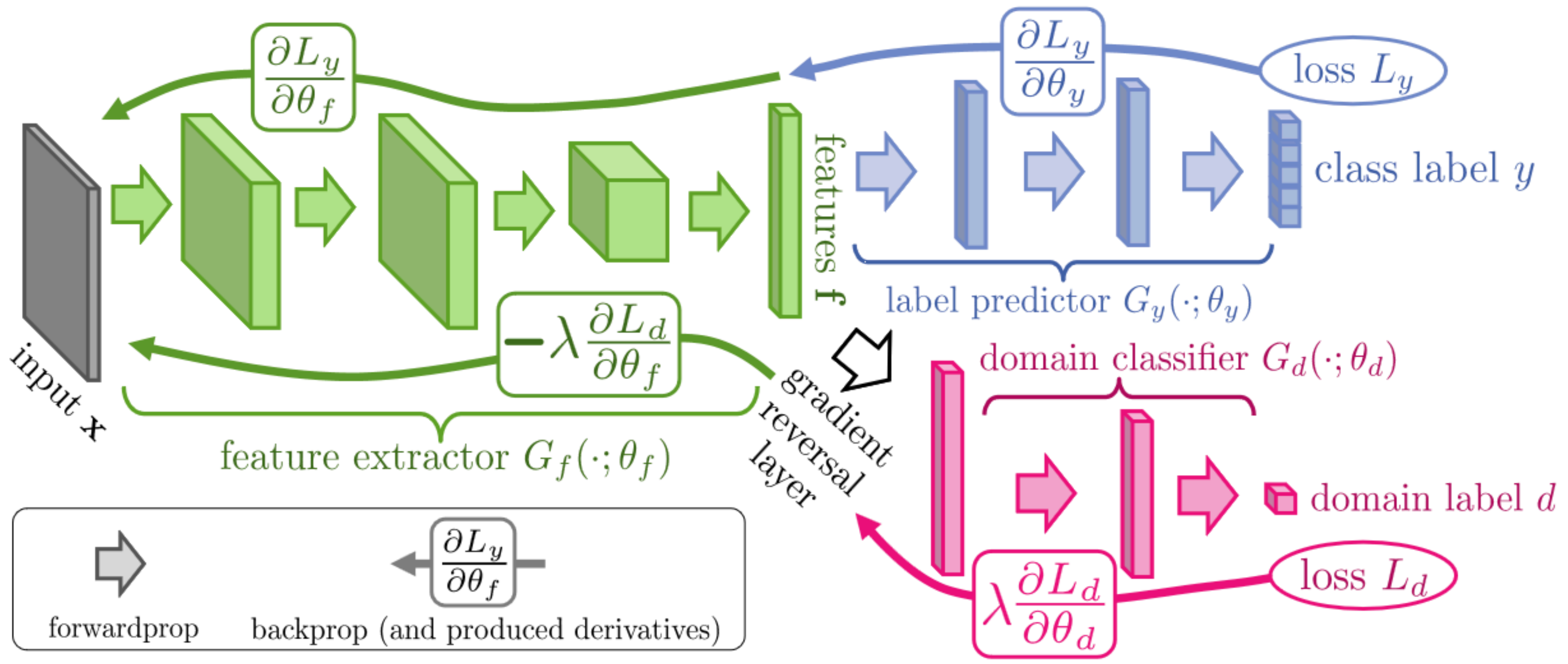
A practical algorithm for domain adaptation

- Learn parameters of all three networks using stochastic gradient updates: gradient descent for G_f and G_y but gradient ascent for G_d!

$$\theta_{f,k+1} = \theta_{f,k} - \alpha\left(\frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f}\right)$$

$$\theta_{y,k+1} = \theta_{y,k} - \alpha \frac{\partial L_y}{\partial \theta_y}$$

- $$\theta_{d,k+1} = \theta_{d,k} - \alpha\lambda \frac{\partial L_d}{\partial \theta_d}$$

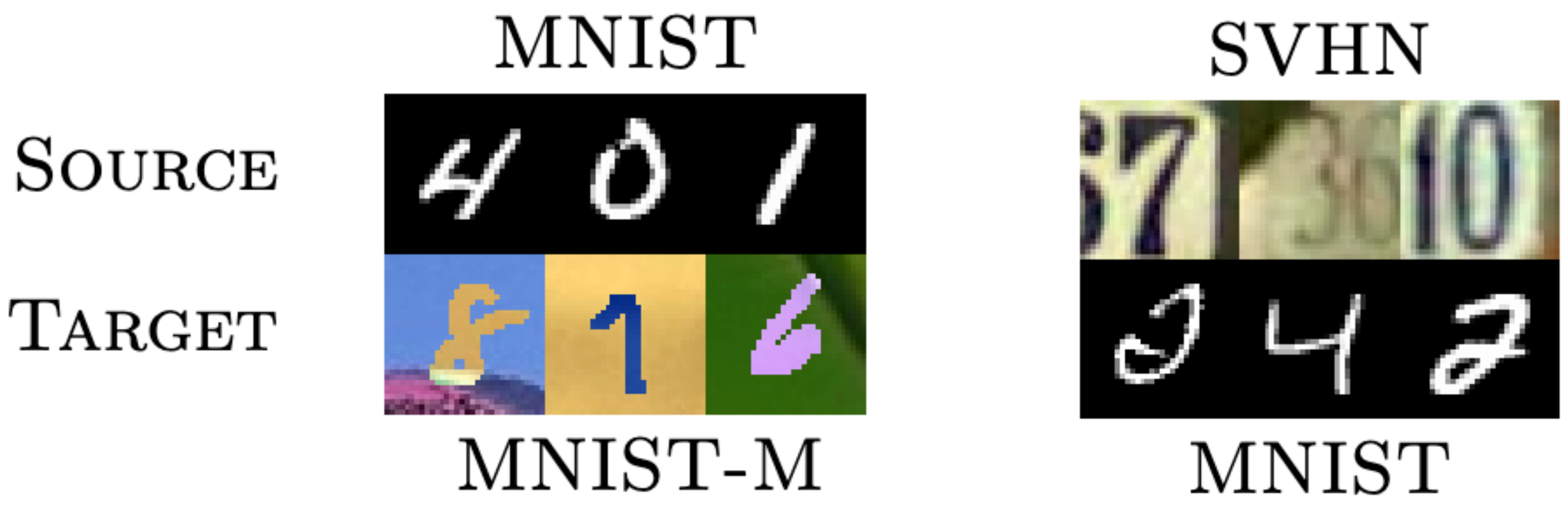


Proposed architecture of the Domain Adversarial Neural Networks (DANN) algorithm

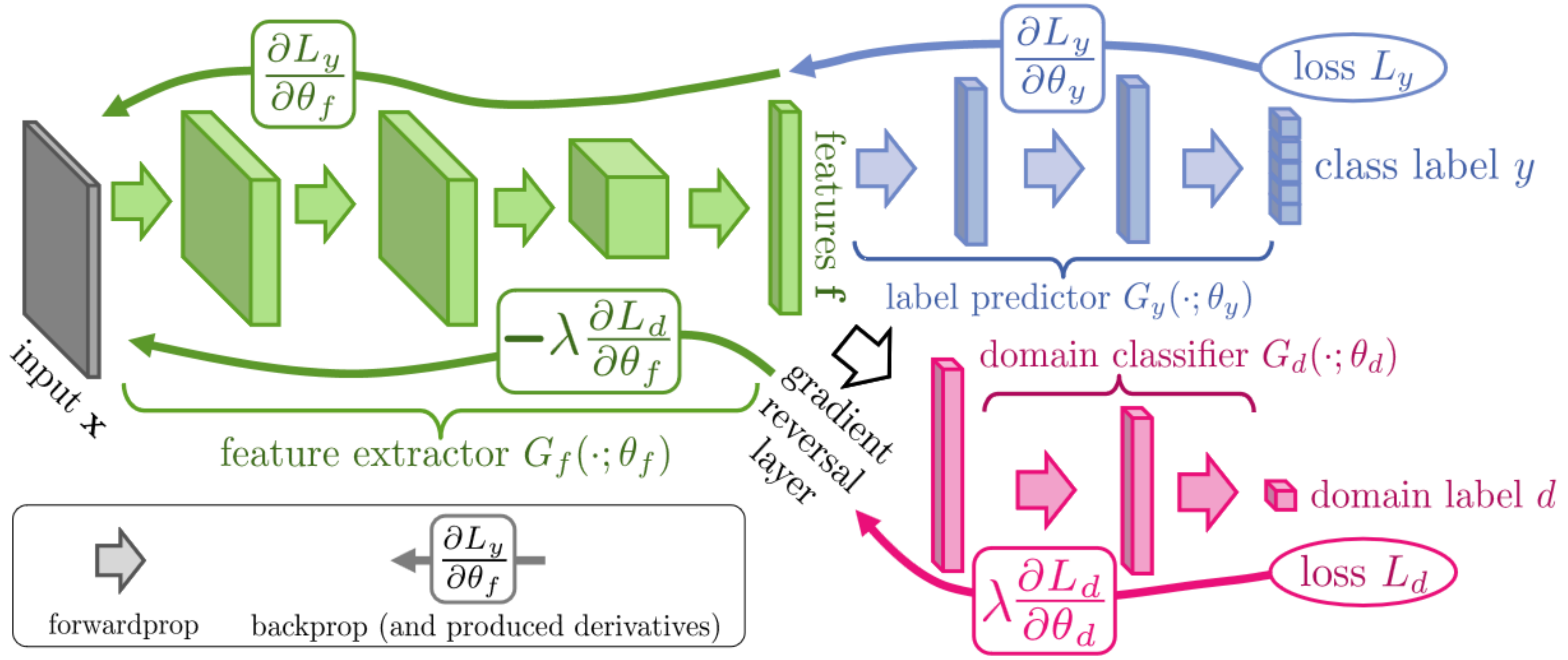
$$\hat{\epsilon}_S(\theta_f, \theta_y, \theta_d) = \frac{1}{N_S}\sum_{i=1}^{N_S} l_y\big(G_y(G_f(x_i, \theta_f), \theta_y), y_i\big)$$
$$-\frac{\lambda}{N_S + N_T}\sum_{i=1}^{N_S+N_T} l_d\big(G_d(G_f(x_i, \theta_f), \theta_d), d_i\big)$$

Supervision signal:
Labels y_i for source
And 1/0 class label d_i for Source/Target mini batch of size N_S, N_T each

Testing DANN on a shifted dataset



- MNIST is shifted to MNIST-M by adding a colored dataset as the background!
- MNIST is compared to SVHN: street-view house number dataset with label as the left-most digit



Proposed architecture of the Domain Adversarial Neural Networks (DANN) algorithm

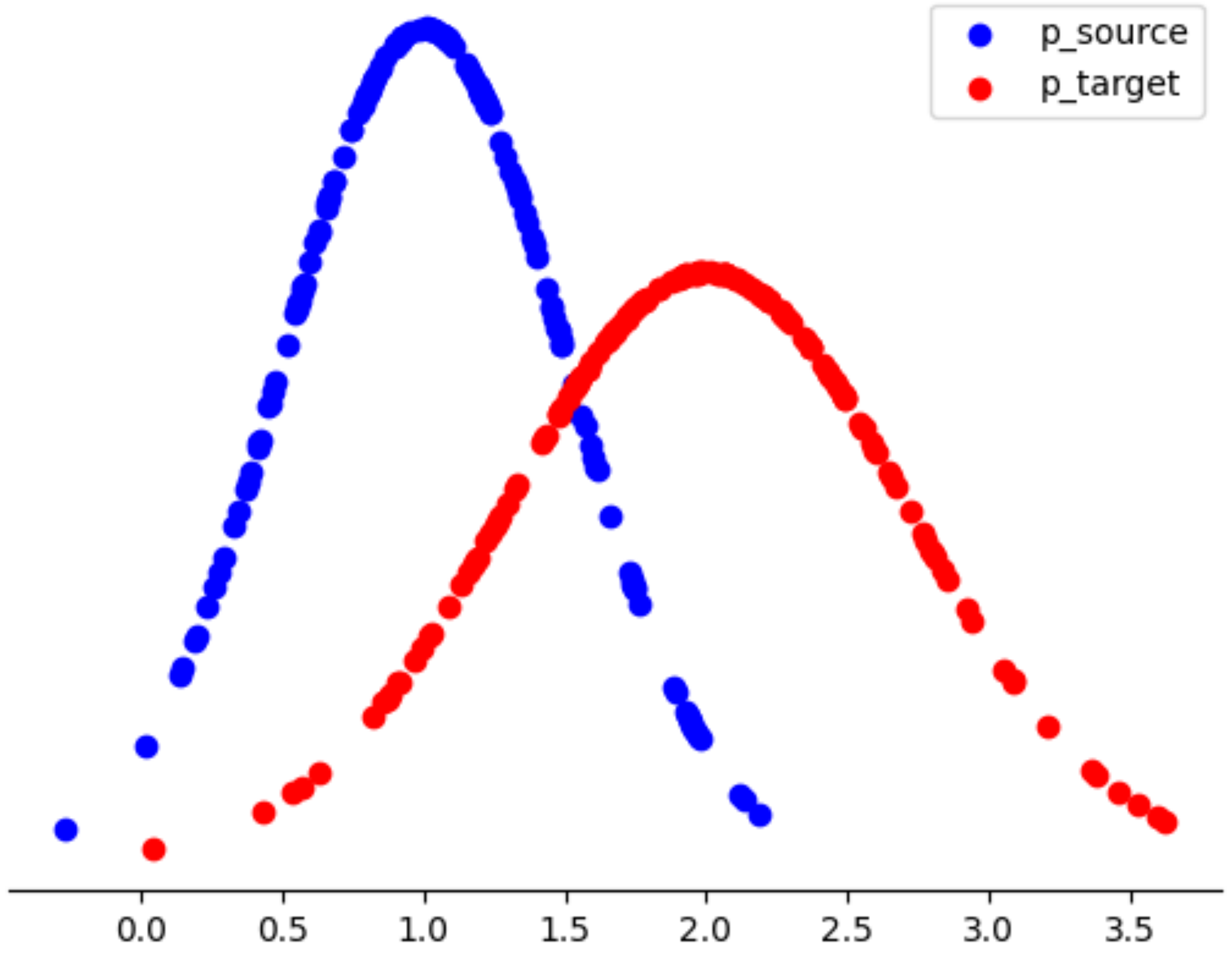
METHOD (SOURCE / TARGET)	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY	.5225	.8674	.5490	.7900
SA (Fernando et al., 2013)	.5690 (4.1%)	.8644 (−5.5%)	.5932 (9.9%)	.8165 (12.7%)
DANN	**.7666** (52.9%)	**.9109** (79.7%)	**.7385** (42.6%)	**.8865** (46.4%)
TRAIN ON TARGET	.9596	.9220	.9942	.9980

Recap: Importance Weighting (IW)

- Prof. Sugiyama introduced IW in first class!
- To get consistent ERM risk, weight the samples using $\frac{p_T(x)}{p_S(x)}$:
- $\mathbb{E}_S\Big[\frac{p_T(x)}{p_S(x)}l(h(x),y)\Big] = \int p_T(y\,|\,x)\frac{p_T(x)}{\cancel{p_S(x)}}l(h(x),y)\cancel{p_S(x)}dxdy = \mathbb{E}_T[l(h(x),y)]$
- Assuming covariate shift (the conditional is invariant $p_S(y\,|\,x) = p_T(y\,|\,x)$)!
- Assumes same support! (Otherwise $\frac{p_T(x)}{p_S(x)}$ estimates are unbounded)

Importance weighting algorithm

- Learn the ratio $w_i \approx \frac{p_T(x_i)}{p_S(x_i)}$ directly from samples using:
    - Kernel mean matching, logistic regression, etc.
    - As opposed to learning both distributions and dividing, which is information theoretically harder, and can result in inflated weights due to division
- In practice bound the weights $w_i < B$ to reduce variance, and require $p_T(x_i) \to 0$ as $p_S(x_i) \to \epsilon, \epsilon > 0$ for numerical stability.
- Then apply weighted ERM: $h_{\mathrm{IW}} = \arg\min_{h \in \mathscr{H}} \underbrace{\sum_{i=1}^{N} w(x_i) l(h(x_i), y_i)}_{=: \hat{\epsilon_{S,w}}}$



When estimating the ratio, as the mass of the target distribution spreads towards outside the source support, the estimation starts becoming numerically unstable

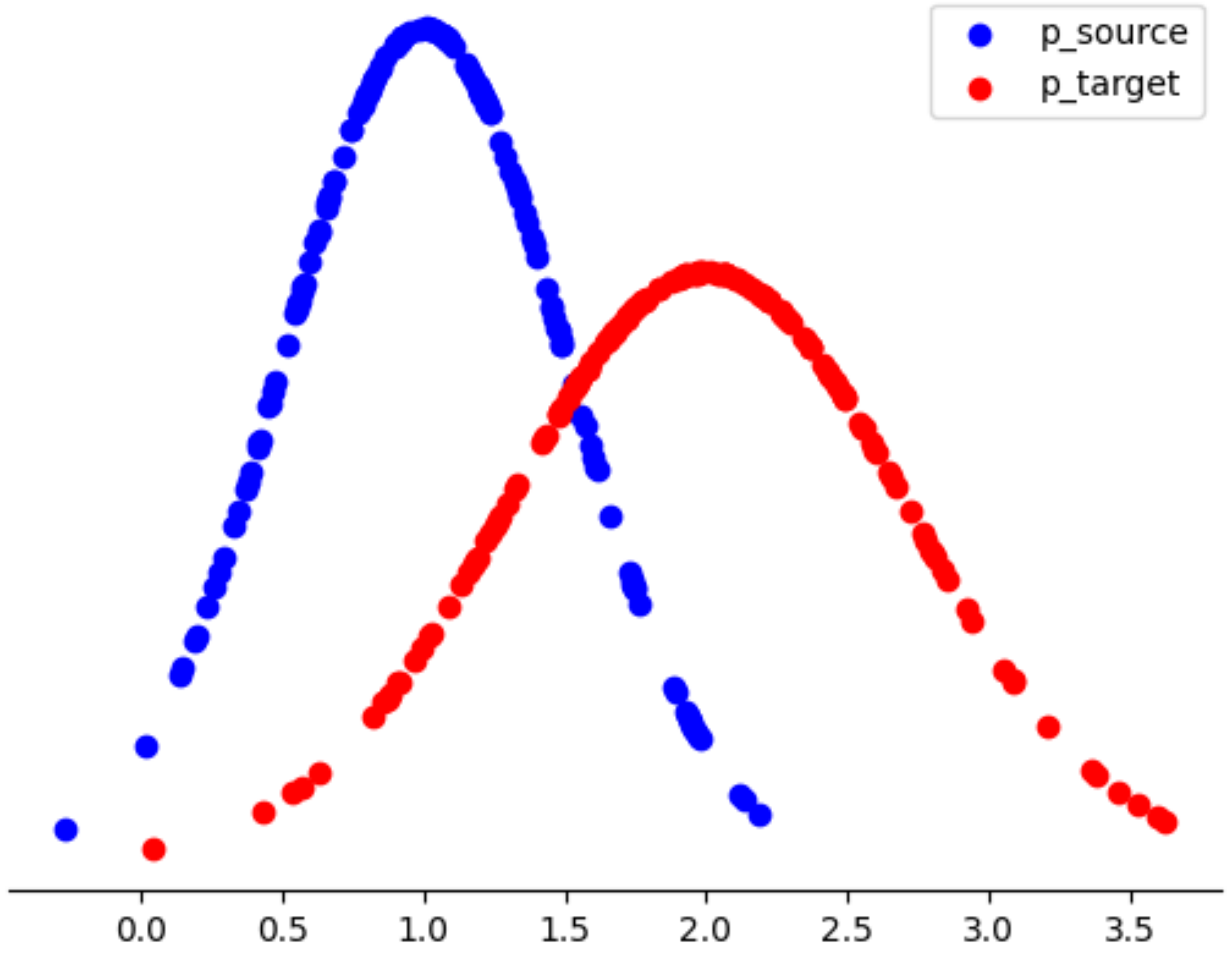
Importance weighting bound

- Consistency is an asymptotic argument, what happens for finite samples?
- Bound shown in [Cortes2010] using Renyi divergences

$$D_\alpha(p_S, p_T) = \frac{1}{\alpha - 1} \log_2 \int p_S(x) \Big(\frac{p_S(x)}{p_T(x)}\Big)^{\alpha-1} dx$$

$$d_\alpha(p_S, p_T) = 2^{D_\alpha(p_S, p_T)}$$

- Extension of KL divergence, $\alpha = 1$ corresponds to KL
- Theorem 3 [Cortes2010]: If $d_2(p_S, p_T)$ is bounded and $w(x) \neq 0$ for all x, w.h.p $1 - \delta$
- $\epsilon_T(h) \leq \hat{\epsilon}_{S,w}(h) + 2^{5/4} \sqrt{d_2(p_S, p_T)} \Big(\frac{d \log 2Ne/d + \log 4/\delta}{N}\Big)^{3/2}$



Bound does not seem to yield a new practical algorithm, as the Renyi divergence here is not easy to estimate and the bound could be very loose!

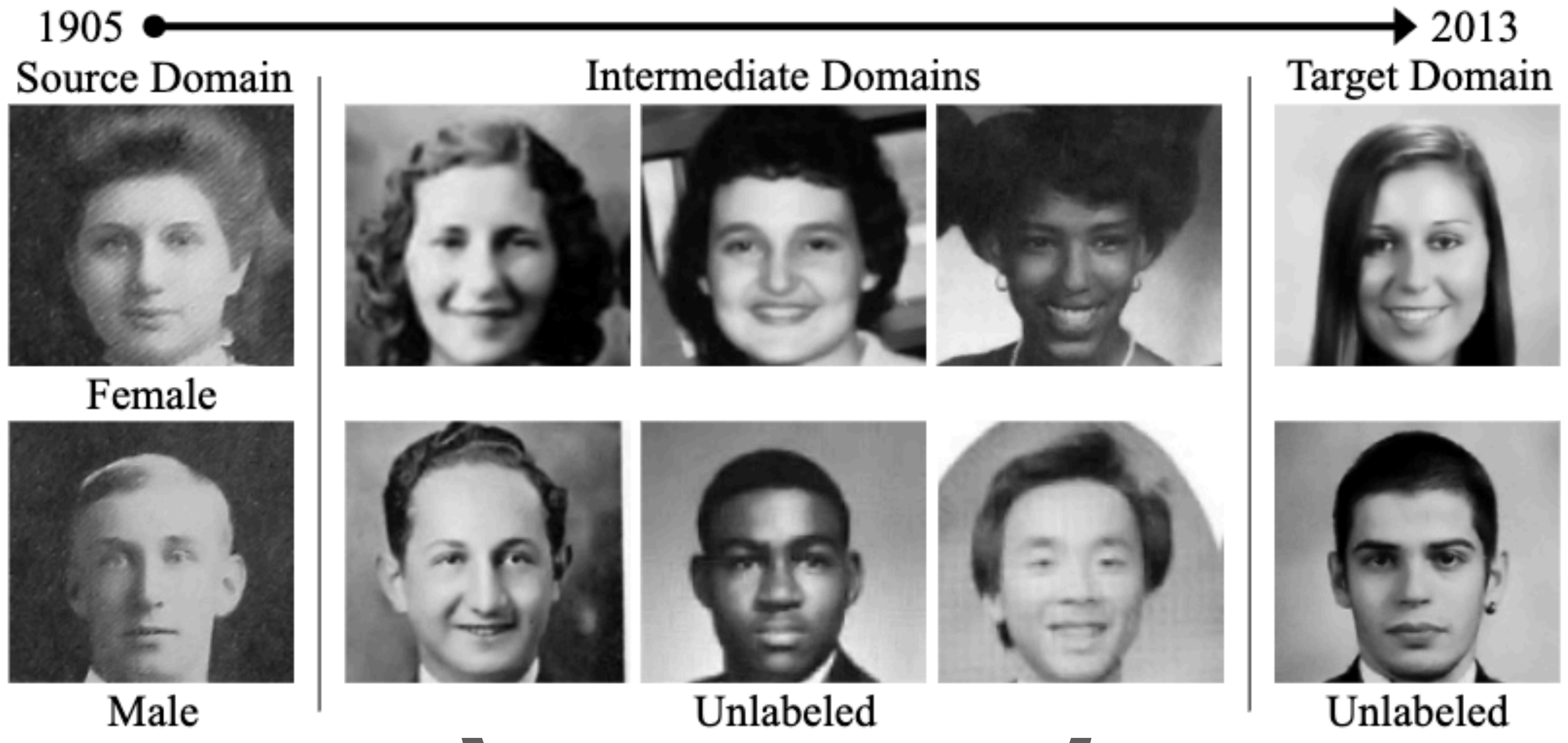
Extensions of domain adaptation

- Static setting: many samples available offline from $p_S(x, y)$ and $p_T(x)$
- Test-time adaptation: at test time source samples may not be available, target samples could be arriving one-batch-at-a-time
- Multi-source adaptation: there are multiple datasets $p_{S_i}(x, y)$ to learn from
- Continual adaptation: there may be a continuous distribution shift over time
- Gradual domain adaptation: shift might be small but accumulating over 'time'

Gradual domain adaptation (GDA)

- GDA: Between the source and the target distributions, there are intermediate datasets available with increasing shift-intensity
- These intermediate data are often indexed ($t = 1, \ldots, D$) but unlabeled
- Portraits dataset as an example
- Pseudo-labeling can be used as a GDA algorithm [Kumar2020, Wang2022]

Portraits: a historical dataset of US high school yearbook images.



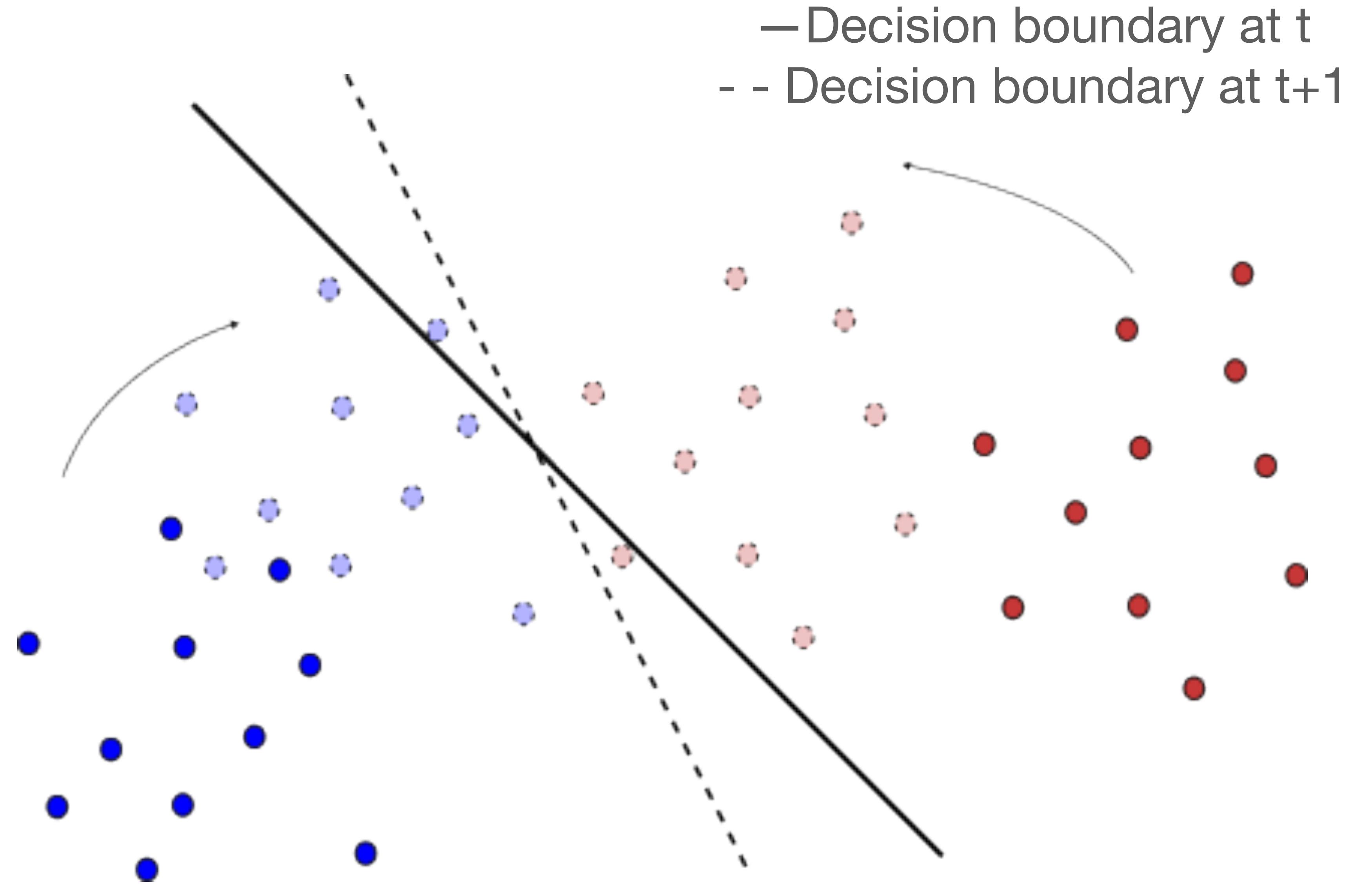
t = 0

t = D+1

Unlabeled indexed datasets from t = 1 to D

Gradual domain adaptation (GDA)

- Pseudo-labeling: an algorithm borrowed from semi-supervised learning literature!
- Self-training (ST) approaches use the labels provided by a modification of the model's predictions
- $h_{t+1} = \text{ST}(h_t, \mathscr{D}_{t+1}) = \arg\min_{f \in \mathscr{H}} \sum_{x \in \mathscr{D}_{t+1}} l(f(x), \tilde{h}_t(x))$
- Where $\tilde{h}_t(x)$ denotes the modified predictions of model at index t, e.g. after converting predictions to one-hot labels



Due to the margin of the current classifier at index t, the modified (e.g. hard) labels of the current classifier are often correct for small shifts, and can be used to train a new classifier with similar margin property

Conclusion

- Robust ML: what we learn should transfer (or degrade gracefully) to other ‘nearby’ problems
    - Domain adaptation belongs to this family of robust ML methods (robust to label noise, adversarial inputs, etc.)
    - We can learn robust features that *generalize* by restricting them to those that appear similarly in both the source and the target distributions
    - We can design algorithms that under certain assumptions can *adapt* successfully to the target (unlabeled) domain.

Code available!

- Some simple code is available: https://github.com/okankoc/lecture_da
  - For slide 7, see `run_pytorch_basic.py`
  - For slide 29, see `run_shift_classification.py`.
    - You can compare importance weighting (IW) with DANN in `run_shift_classification.py`

Homework

- Homework: write a 2 page essay describing the domain adaptation problem, giving examples from real-world examples where it can make an impact. What are some of the challenges? Do you have any suggestions to resolve these issues?

- Alternative: run the script `run_shift_classification.py` and report your findings. Change the model, change the parameters of the algorithms, what happens?

References

- SBD2010: Shai Ben-David et al. A theory of learning from different domains, Machine Learning (2010), vol. 79, pg. 151-175
- Mansour2009: Mansour et al. Domain Adaptation: Learning Bounds and Algorithms, 2009
- DANN: Ganin et al. Domain-Adversarial Training of Neural Networks, JMLR vol.17 (2016), pg. 1-35
- Cortes2010: Cortes et al. Learning Bounds for Importance Weighting, NeurIPS 2010

References

- CORAL2016: Sun et al. Correlation Alignment for Unsupervised Domain Adaptation
- BN2021: Nado et al. Evaluating Prediction-Time Batch Normalization for Robustness under Covariate Shift
- Kumar2020: Kumar et al. Understanding Self-Training for Gradual Domain Adaptation, ICML 2020
- Wang2022: Wang et al. Understanding Gradual Domain Adaptation: Improved Analysis, Optimal Path and Beyond, ICML 2022

Open questions

- Computable / fully optimizable (and tight!) upper bounds for domain adaptation (DA)
    - Under which (reasonable) assumptions?
    - If we need labels, can we compute tight bounds with very few labels?
    - How does (density of) model misspecification play a role?
- Many extensions possible (as partly discussed in slide 33)
    - Test-time DA: without access to training inputs, how can we do the adaptation?