CSE 291: Domain Adaptation in Vision
Manmohan Chandraker

Lecture 2: Easy Adaptation
Course details

- **Grading**
  - 20% presentation
  - 10% reviews
  - 40% project
  - 20% final exam
  - 10% participation

- **Aim is to learn together, discuss and have fun!**
Course details

• Grading
  – 20% presentation
  – 10% reviews
  – 40% project
  – 20% final exam
  – 10% participation

• Aim is to learn together, discuss and have fun!

Do you want to have a final exam? closes in 2 day(s)

A total of 27 vote(s) in 118 hours

- 6 (22% of users) | I want to have a final exam.
- 16 (59% of users) | I don't want to have a final exam.
- 5 (19% of users) | Either one works for me.
Course details

• **Presentation instructions**
  – 1-2 students to present in each class
  – Discuss topic with instructor one week in advance
  – Send slides by 9pm two days before the class
  – Allow speaking time of 20 minutes each (about 15 slides)
  – Presentation should be well-organized and thoughtful
  – Ask questions and encourage discussions along the way

• Each student does one presentation
Course details

• Presentation contents
  – Summarize the topic and how the papers address it
  – Why the topic is interesting, or difficulty of the problem
  – Motivate with applications
  – Key technical ideas, why they are interesting
  – Strengths and weakness of proposed methods
  – Detailed analysis of experiments
  – If possible, include own analysis based on author code
  – Open problems, extensions, likely follow-up papers
Course details

• Reviews
  – Day before class, send a brief review of 1 paper
  – Upload on Gradescope

• Review format (follow exactly)
  1. Summary of the paper (3-4 sentences)
  2. Strengths
  3. Weaknesses
  4. Critique of experiments
  5. Possible extensions or follow-ups (come up with at least one)

• Presenters need not send in review for that class
• Ask questions, answer them, engage in discussions
Papers for Wed, Jan 22

• Domain Adaptation via Transfer Component Analysis

• Learning Transferable Features with Deep Adaptation Networks
Recap
Goals of computer vision

• Challenges:
  • Large amounts of labeled data is needed
  • Many tasks do not have large-scale labeled data
  • Some tasks or data change over time
  • There exist too many tasks

• Solution: domain adaptation
  • Generalization of knowledge learned in previous domains or tasks, to new domains or tasks
Train on Source, Test on Target

Challenge of domain adaptation:
- Labels only in source domain, classification conducted in target domain
- Classifier trained in source not applicable in target, due to distribution shift
Different types of data shift

- Data distributions $p(x, y)$ cannot change in arbitrary ways
  - Example: call “cats” as “dogs” and the other way round
- Adaptation possible under reasonable assumptions of data shift
Covariate Shift

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>cat</td>
</tr>
<tr>
<td>cat</td>
<td>dog</td>
</tr>
<tr>
<td>dog</td>
<td>dog</td>
</tr>
</tbody>
</table>

• Input *data distributions* change, but the labeling function is unchanged
• $P^s(X) \neq P^t(x)$, but $P^s(y \mid x) = P^t(y \mid x)$
• Reasonable assumption when we believe $x$ causes $y$
• S: British, T: American, $x$: the event that it rains, $y$: talking about weather
Label Shift

- Marginal distribution over labels changes, $P^s(y) \neq P^t(y)$
- Class conditionals remain unchanged, $P^s(x \mid y) = P^t(x \mid y)$
- Reasonable assumption when we believe $y$ causes $x$
- Example: $y$ is a diagnosis, $x$ is the symptom
- Typically, when classes unbalanced across source and domain

![Prior Probability Shift](image)
Concept Shift

- *Label definitions* change, $P_s(y \mid x) \neq P_t(y \mid x)$
- Example: $x$ is \{soda, pop, coke\}, $y$ is whether it is soft drink
- Can assume: $P(y \mid x)$ changes slowly across domains
Common issue: Abundant labeled data in some domain, but none for application domain.

General Approaches for Wide Variety of Tasks

Labeled:
- Good weather, daylight
- Still images, frontal, unoccluded
- Internet domain

Unlabeled:
- Rainy weather, low light
- Video, motion, side view, occlusions
- Surveillance domain
Types of Approaches

• Domain invariant feature learning
• Domain mapping
• Normalization
• Ensembling
• Instance reweighting
• Domain generalization
Domain Invariance

- Align source and target by learning a *domain invariant* feature
  - Feature follows same distribution whether from source or target
  - Source classifier may then generalize well to target domain
  - Assumes such a representation exists
  - Assumes marginal label distributions similar: $P^s(y) = P^t(y)$
Domain Mapping

- Translate source image at pixel-level, to resemble image from target distribution
- Train classifier using labels for translated source images

Real examples

Goal of $D$: classify as real,
$D(x) = 1$

Generated examples

Goal of $D$: classify as not real,
$D(G(z)) = 0$

Competing goal of $G$: $D$ misclassify as real, $D(G(z)) = 1$
Domain Mapping

Training

Testing
Domain Mapping

Training

Testing
Normalization

- Batch norm: normalize mean and standard deviation for each feature channel
  - Ensure each layer receives data from a similar distribution

- AdaBN: transfer model to new domain by modulated batch norm statistics
  - Layer weights store task-specific knowledge
  - Batch norm statistics learn domain-specific knowledge

- No parameters to tune, easy to compute, complementary to other methods
Ensembling

- Train independent source classifiers
- Different augmentations or dropout for diversity
- Pseudo-label target examples if source classifiers agree on prediction
- Train target classifier on pseudo-labeled examples
Domain Generalization
Many Flavors of Domain Invariance

Divergence
- MMD
- Correlation
- Contrastive
- Wasserstein

Adversarial
- Generative
- Non-generative

Reconstruction
- Encoder-Decoder
- GAN

Deep domain confusion (DDC)
Deep adaptation network (DAN)
Joint adaptation network (JAN)
Residual transfer network (RTN)
Generative adversarial (GAN)
DANN
ADDA
Feature augmentation
Reconstruction classification (DRCN)
Domain separation (DSN)
Deep Domain Adaptation

Learn a representation to minimize discrepancy
Deep Domain Adaptation

Learn a representation to minimize discrepancy
Deep Domain Adaptation

Learn a representation to minimize discrepancy
Adversarial Approach to Domain Adaptation

**Traditional Training**

- Training dataset
  - Source images (known labels)
- Feature layers
- Task classifier
- Labels
- Testing
  - Source
  - Target
  - Performs well on source, but not on target images

**Domain Adversarial Training**

- Training dataset
  - Source images (known labels)
  - Target images (unknown labels)
- Feature layers
- Task classifier
- Domain discriminator
- Labels
- Testing
  - Source
  - Target
  - Classify as source or target
  - Performs well on source, as well as target images
  - Good
Theory
Generalization guarantee

- $\text{Risk} \leq \text{SomethingControllable} + \text{SomethingSmall}$
- **Risk** is the performance on unseen data, i.e. the performance on the test set
- **SomethingControllable**, e.g. the performance measured on the training set or the performance of an hypothesis on the source task
- **SomethingSmall** depends on
  - The number of samples
  - The confidence (it is a probabilistic bound)
  - Etc.
- For example, the relation between the tasks can appear in either terms
Issues specific to domain adaptation

In addition to the usual ML issues, here we also have the following problems:

• **When does DA/TL work?**
  – In other words, when it is impossible to have a proper transfer of knowledge?

• **How to do model selection without enough information?**
  – How to tune hyperparameters with no labeled information from the target task (and no tuning on the test set)?
Different theoretical settings

- **Conservative Case**
  - Learner finds the best hypothesis with respect to the labeled training source sample. The target unlabeled sample are not used

- **Non Conservative Case**
  - Learner utilizes the information contained in the target domain sample in the process of choosing the classifier

- **Hypothesis Transfer Learning**
  - No access to source sample, but only to source hypotheses
Example: H-divergence

Domain classifier: \( h(x) = 0 \) for source and \( h(x) = 1 \) for target samples

\[
d_H(D_S, D_T) = 2 \sup_{h \in \mathcal{H}} \left| \Pr_{x \sim D_S} [h(x) = 1] - \Pr_{x \sim D_T} [h(x) = 1] \right|
\]

(discrepancy)  
(Error of the best domain classifier)

Target error bounded by source error and domain discrepancy

\[
\epsilon_T(h) \leq \epsilon_S(h) + d_H(D_S, D_T) + \lambda
\]

Adaptability: Some model should achieve low error on both source and target
Adversarial Domain Adaptation
Instance Reweighting
Covariate Shift

- Source domain $S$, target domain $T$
- Labeled examples $\{(x, y)\}$ in $S$, only unlabeled examples $\{x\}$ in $T$
- Goal: find classifier $h(x, \theta)$ to minimize the expected error in $T$

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{p(x,y)}[h(x, \theta) \neq y]$$
Covariate Shift

• Source domain $S$, target domain $T$
• Labeled examples $\{(x, y)\}$ in $S$, only unlabeled examples $\{x\}$ in $T$
• Goal: find classifier $h(x, \theta)$ to minimize the expected error in $T$

$$\theta^* = \arg \min E_{P(x,y)} [h(x, \theta) \neq y]$$

• Covariate shift assumption:
  $$P_S(X) \neq P_T(X), \text{ but } P_S(Y|X = x) = P_T(Y|X = x)$$
  – Marginal distributions are not the same
  – Distribution of emails for Alice might be different from those for Bob
  – Conditional distributions are the same
  – Given a specific email, probability of being spam same for Alice and Bob

• Not always satisfied, but often still a useful approximation
Model Misspecification

• Covariate shift is problematic for misspecified models
  – For a classifier, the only output we care about is $P_T(Y|X = x)$
  – Since $P_S(Y|X = x) = P_T(Y|X = x)$, simply train on $S$ and test on $T$
Model Misspecification

• Covariate shift is problematic for misspecified models
  – For a classifier, the only output we care about is $P_T(Y|X = x)$
  – Since $P_S(Y|X = x) = P_T(Y|X = x)$, simply train on S and test on T
  – Would be fine if we had a perfect model
  – But almost always, the function we fit is not what generated the data

Logistic regression to fit a straight line is a clearly misspecified model here.
Model Misspecification in Adaptation

• Consider labeled source data and unlabeled target data
  – Clearly, target data has higher $x_1$ values, in general
  – Thus, $P_S(X) \neq P_T(X)$.

• Suppose we wish to train a classifier for target domain
  – We choose to fit a linear classifier with logistic regression
Model Misspecification in Adaptation

- Fit a linear classifier to source examples using logistic regression
  - Seems quite good for source data
  - Clearly sub-optimal for target data
- Not possible to find parameter $\theta$ such that, for all $x$,
  \[
P_S(Y|X = x, \theta) = P_T(Y|X = x, \theta)
  \]
  - Model misspecification: optimal source and target models are different.
Risk Minimization

- True risk minimization
  - Minimize expected value of loss over true joint distribution of data $P(x, y)$

$$
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_P [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int_{y \in \mathcal{Y} \ x \in \mathcal{X}} P(x, y) l(x, y, \theta) \, dx \, dy
$$
Risk Minimization

• True risk minimization
  – Minimize expected value of loss over true joint distribution of data $P(x, y)$

$$
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_P [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int P(x, y) l(x, y, \theta) \, dx \, dy
$$

• Empirical risk minimization
  – True joint distribution of data is unknown
  – Approximate with empirical distribution, by sampling $(x_i, y_i) \sim P(x, y)$.

$$
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_P [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int P(x, y) l(x, y, \theta) \, dx \, dy
\approx \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{i \in \{1, \ldots, N\}} l(x_i, y_i, \theta)
= \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} l(x_i, y_i, \theta)
$$
Reweighting for Domain Adaptation

- We now have two distributions: $P_S(x, y)$ and $P_T(x, y)$
  - Train on empirical source distribution, but optimize risk for target
Reweighting for Domain Adaptation

- We now have two distributions: $P_S(x, y)$ and $P_T(x, y)$
  - Train on empirical source distribution, but optimize risk for target

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{P_T}[l(x, y, \theta)] = \arg\min_{\theta \in \Theta} \int_{y \in \mathcal{Y}} \int_{x \in \mathcal{X}} P_T(x, y) l(x, y, \theta) \, dx \, dy$$
Reweighting for Domain Adaptation

- We now have two distributions: $P_S(x, y)$ and $P_T(x, y)$
  - Train on empirical source distribution, but optimize risk for target

\[
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T} [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int_{y \in Y \; x \in X} P_T(x, y) l(x, y, \theta) \, dx \, dy
\]

\[
= \arg \min_{\theta \in \Theta} \int \int_{y \in Y \; x \in X} \frac{P_T(x, y)}{P_S(x, y)} P_S(x, y) l(x, y, \theta) \, dx \, dy
\]
Reweighting for Domain Adaptation

• We now have two distributions: $P_S(x, y)$ and $P_T(x, y)$
  - Train on empirical source distribution, but optimize risk for target

$$
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T}[l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int_{x \in \mathcal{X}, y \in \mathcal{Y}} P_T(x, y) l(x, y, \theta) \, dx \, dy
$$

$$
= \arg \min_{\theta \in \Theta} \int \int_{x \in \mathcal{X}, y \in \mathcal{Y}} \frac{P_T(x, y)}{P_S(x, y)} P_S(x, y) l(x, y, \theta) \, dx \, dy
$$

Sample $(x_i, y_i) \sim P_S(x, y) \quad \approx \quad \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i, y_i)}{P_S(x_i, y_i)} l(x_i, y_i, \theta)$
Reweighting for Domain Adaptation

- We now have two distributions: \( P_S(x, y) \) and \( P_T(x, y) \)
  
  - Train on empirical source distribution, but optimize risk for target

\[
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T} [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \int \int P_T(x, y) l(x, y, \theta) \, dx \, dy
\]

\[
= \arg \min_{\theta \in \Theta} \int \int \frac{P_T(x, y)}{P_S(x, y)} P_S(x, y) l(x, y, \theta) \, dx \, dy
\]

Sample \((x_i, y_i) \sim P_S(x, y)\)  

By definition of joint probability  

\[
\approx \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i, y_i)}{P_S(x_i, y_i)} l(x_i, y_i, \theta)
\]

\[
= \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i) P_T(y_i|x_i)}{P_S(x_i) P_T(y_i|x_i)} l(x_i, y_i, \theta)
\]
Reweighting for Domain Adaptation

- We now have two distributions: $P_S(x, y)$ and $P_T(x, y)$
  - Train on empirical source distribution, but optimize risk for target

$$
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T} [l(x, y, \theta)]
 = \arg \min_{\theta \in \Theta} \int \int_{y \in \mathcal{Y}, x \in \mathcal{X}} P_T(x, y) l(x, y, \theta) \, dx \, dy

= \arg \min_{\theta \in \Theta} \int \int_{y \in \mathcal{Y}, x \in \mathcal{X}} \frac{P_T(x, y)}{P_S(x, y)} P_S(x, y) l(x, y, \theta) \, dx \, dy
$$

Sample $(x_i, y_i) \sim P_S(x, y) \quad \Rightarrow \quad \approx \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i, y_i)}{P_S(x_i, y_i)} l(x_i, y_i, \theta)$

By definition of joint probability $\Rightarrow \quad = \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i) P_T(y_i|x_i)}{P_S(x_i) P_S(y_i|x_i)} l(x_i, y_i, \theta)$

Covariate shift assumption $\Rightarrow \quad = \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i)}{P_S(x_i)} l(x_i, y_i, \theta)$

$P_S(Y|X = x) = P_T(Y|X = x)$ for all $x$
Reweighting for Domain Adaptation

• Can approximate *true risk under target* distribution
  – Using *empirical risk under source* distribution
  – By reweighting source samples \((x_i, y_i) \sim P_S(x, y)\)

\[
\theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T}[l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \sum_{i=1,...,N} \frac{P_T(x_i)}{P_S(x_i)} l(x_i, y_i, \theta)
\]

– Reweight each sample using ratio of marginal covariate probabilities \(\frac{P_T(x_i)}{P_S(x_i)}\)

– Hidden assumption:
Reweighting for Domain Adaptation

- Can approximate true risk under target distribution
  - Using empirical risk under source distribution
  - By reweighting source samples \((x_i, y_i) \sim P_S(x, y)\)

\[
\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{P_T}[l(x, y, \theta)] = \arg\min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i)}{P_S(x_i)} l(x_i, y_i, \theta)
\]

- Reweight each sample using ratio of marginal covariate probabilities \(\frac{P_T(x_i)}{P_S(x_i)}\)

- Hidden assumption: support of \(P_T(x)\) is contained within support of \(P_S(x)\)
Reweighting for Domain Adaptation

- Can approximate *true risk under target* distribution
  - Using *empirical risk under source* distribution
  - By reweighting source samples \((x_i, y_i) \sim P_S(x, y)\)
    \[
    \theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{P_T} [l(x, y, \theta)] = \arg \min_{\theta \in \Theta} \sum_{i \in \{1, \ldots, N\}} \frac{P_T(x_i)}{P_S(x_i)} l(x_i, y_i, \theta)
    \]
  - Reweight each sample using ratio of marginal covariate probabilities
  - Hidden assumption: support of \(P_T(x)\) is contained within support of \(P_S(x)\)

- Challenge:
  - Marginal distributions \(P_T(x)\) and \(P_S(x)\) can be difficult to determine
Estimating Marginal Probability Ratio

• Let $N_S$ be size of source data, $N_T$ size of target data
• Instead of directly computing marginals, we can interpret ratio as

$$\frac{P_T(x_i)}{P_S(x_i)} \approx \frac{N_S}{N_T} \frac{P(x_i \text{ comes from the target data})}{P(x_i \text{ comes from the source data})}$$
Estimating Marginal Probability Ratio

• Let $N_S$ be size of source data, $N_T$ size of target data
• Instead of directly computing marginals, we can interpret ratio as

$$\frac{P_T(x_i)}{P_S(x_i)} \approx \frac{N_S}{N_T} \frac{P(x_i \text{ comes from the target data})}{P(x_i \text{ comes from the source data})}$$

  – Draw random samples of size $N_S$ and $N_T$ from source and target distributions
  – Draw an instance $x_i$ from from merged dataset of size $N_S + N_T$
  – Let $n_i^S$ be number of instance of $x_i$ in source samples, $n_i^T$ in target

$$P(x_i \text{ comes from the target data}) = \frac{n_i^T}{n_i^T + n_i^S}$$
Estimating Marginal Probability Ratio

- Let $N_S$ be size of source data, $N_T$ size of target data
- Instead of directly computing marginals, we can interpret ratio as

\[
\frac{P_T(x_i)}{P_S(x_i)} \approx \frac{N_S}{N_T} \frac{P(x_i \text{comes from the target data})}{P(x_i \text{comes from the source data})}
\]

  - Draw random samples of size $N_S$ and $N_T$ from source and target distributions
  - Draw an instance $x_i$ from from merged dataset of size $N_S + N_T$
  - Let $n_i^S$ be number of instance of $x_i$ in source samples, $n_i^T$ in target

\[
P(x_i \text{comes from the target data}) = \frac{n_i^T}{n_i^T + n_i^S}
\]

  - We can now obtain

\[
\mathbb{E} \left[ \frac{P(x_i \text{comes from the target data})}{P(x_i \text{comes from the source data})} \right] = \mathbb{E} \left[ \frac{n_i^T}{n_i^S + n_i^T} \right] = \mathbb{E} \left[ \frac{n_i^T}{n_i^S} \right] = \frac{P_T(x_i)N_T}{P_S(x_i)N_S}
\]

- For each target instance, need to estimate probability it originated from source distribution
Domain Classification for Reweighting

• A *domain classifier* that can give probabilities of a sample being drawn from source or target distributions

• Compute source instance weights
  – Train a logistic regression classifier separating source from target
  – Apply classifier to each source instance $x_i^S$, which yields
    \[ p_i = P(x_i^S \text{ comes from the target data}) \]
    – For each source instance, compute the instance weight as $w_i = \frac{p_i}{1-p_i}$

• Train classifier on *reweighted* source data, separating class 1 and 2
• Apply classifier on *unlabeled* target data
Domain Adaptation by Instance Reweighting
Feature Augmentation
(Supervised)
Frustratingly Easy Adaptation

- Labeled data available in source and (limited) target domain
- **Baseline 1**: train models on individual domains
  - Does not leverage common patterns across domains
- **Baseline 2**: train model on union of data from both domains
  - Does not preserve domain-specific patterns
  - Target data can be “swamped” by larger source data
Frustratingly Easy Adaptation

• Labeled data available in source and (limited) target domain

• Baseline 1: train models on individual domains
  – Does not leverage common patterns across domains

• Baseline 2: train model on union of data from both domains
  – Does not preserve domain-specific patterns
  – Target data can be “swamped” by larger source data

• Easy adaptation:

  Let $\mathcal{X} = \mathbb{R}^F$ be $F$-dimensional feature space.

  Define $\Phi^s, \Phi^t : \mathbb{R}^F \rightarrow \mathbb{R}^{3F}$
  
  $\Phi^s : x \mapsto \langle x, x, 0 \rangle$
  $\Phi^t : x \mapsto \langle x, 0, x \rangle$

  Train classifier in augmented space.
Frustratingly Easy Adaptation

- Task: word classification (noun, verb, determiner)
- Source domain: WSJ, target domain: Wired
  - “the” is a determiner in both source and target
  - “monitor” is likely verb in source, but noun in target
Frustratingly Easy Adaptation

• Task: word classification (noun, verb, determiner)
• Source domain: WSJ, target domain: Wired
  – “the” is a determiner in both source and target
  – “monitor” is likely verb in source, but noun in target

Original features \((x_1, x_2)\): \(x_1\) is indicator for “the”, \(x_2\) for “monitor”

Augmented features \((x_1, x_2, \tilde{x}_1^s, \tilde{x}_2^s, \tilde{x}_1^t, \tilde{x}_2^t)\)

Weights: \((w_1, w_2, \tilde{w}_1^s, \tilde{w}_2^s, \tilde{w}_1^t, \tilde{w}_2^t)\)

\(w_1, w_2\): general feature weights for “the”, “monitor”

\(\tilde{w}_1^s, \tilde{w}_2^s\): source domain features

\(\tilde{w}_1^t, \tilde{w}_2^t\): target domain features
Frustratingly Easy Adaptation

- Task: word classification (noun, verb, determiner)
- Source domain: WSJ, target domain: Wired
  - “the” is a determiner in both source and target
  - “monitor” is likely verb in source, but noun in target

Original features \((x_1, x_2)\): \(x_1\) is indicator for “the”, \(x_2\) for “monitor”

Augmented features \((x_1, x_2, \tilde{x}_1^s, \tilde{x}_2^s, \tilde{x}_1^t, \tilde{x}_2^t)\)

Weights: \((w_1, w_2, \tilde{w}_1^s, \tilde{w}_2^s, \tilde{w}_1^t, \tilde{w}_2^t)\)

- \(w_1, w_2\): general feature weights for “the”, “monitor”
- \(\tilde{w}_1^s, \tilde{w}_2^s\): source domain features
- \(\tilde{w}_1^t, \tilde{w}_2^t\): target domain features

Example weights that might be learned for various classes
- Determiner: \((1, 0, 0, 0, 0, 0)\), so “the” is determiner in both source and target
- Verb: \((0, 0, 0, 1, 0, 0)\), so “monitor” is verb only in source
- Noun: \((0, 0, 0, 0, 1)\), so “monitor” is noun only in target