Machine Learning
Data Analytics

FAU

Machine Learning for Time Series (MLTS)

Lecture 14: MLTS in the Real World

Part 2: Domain Adaptation

Dr. Dario Zanca

Machine Learning and Data Analytics (MaD) Lab
Friedrich-Alexander-Universität Erlangen-Nürnberg
19.12.2024

Topics overview

1. Time series fundamentals and definitions (Part 1)
2. Time series fundamentals and definitions (Part 2)
3. Bayesian Inference and Gaussian Processes
4. State space models (Kalman Filters)
5. State space models (Particle Filters)
6. Autoregressive models
7. Data mining on time series
8. Deep Learning (DL) for Time Series (Introduction to DL)
9. DL – Convolutional models (CNNs)
10. DL – Recurrent models (RNNs and LSTMs)
11. DL – Attention-based models (Transformers)
12. DL – From BERT to ChatGPT
13. DL – New Trends in Time Series processing
14. Time series in the real world

Topics overview

1. Time series fundamentals and definitions (Part 1)
2. Time series fundamentals and definitions (Part 2)
3. Bayesian Inference and Gaussian Processes
4. State space models (Kalman Filters)
5. State space models (Particle Filters)
6. Autoregressive models
7. Data mining on time series
8. Deep Learning (DL) for Time Series (Introduction to DL)
9. DL – Convolutional models (CNNs)
10. DL – Recurrent models (RNNs and LSTMs)
11. DL – Attention-based models (Transformers)
12. DL – From BERT to ChatGPT
13. DL – New Trends in Time Series processing
14. Time series in the real world

In this lecture…

1. *Domain adaptation: overview*
2. *Unsupervised domain adaptation*
3. *Domain generalization (OOD generalization)*
4. *Recap*

References

Largely based on “Deep Learning Foundations” course by Soheil Feizi (University of Maryland):

- https://www.youtube.com/watch?v=El760ZzsXN8
- https://www.youtube.com/watch?v=wwgt_ErD3vA

Domain adaptation

Domain adaptation: overview

The typical machine learning setup (so far)

The typical setup we have had so far included a training set

$$\{(x_i^{train}, y_i^{train})\}_{i=1}^{m} \sim Q_{X,Y}$$

Where $x_i \in X$, $y_i \in Y$, and where $Q_{X,Y}$ denotes the distribution from which the training examples are sampled from.

Again, typically we want to learn an optimal mapping f_θ, for which we solve:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(f_\theta(x_i^{train}), y_i^{train}) \Rightarrow \theta^*$$

The typical machine learning setup (so far)

We, then, evaluate our model on a hold out test set

$$\{(x_i^{test}, y_i^{test})\}_{i=1}^{m'} \sim Q_{X,Y}$$

by computing a test error

$$\epsilon_{test} = \frac{1}{m} \sum_{i=1}^{m'} L(f_{\theta^*}(x_i^{test}), y_i^{test})$$

(and we aim at a small ϵ_{test}).

The typical machine learning setup (so far)

Summary:

1. $\{(x_i^{train}, y_i^{train})\}_{i=1}^{m} \sim Q_{X,Y}$
2. $\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(f_\theta(x_i^{train}), y_i^{train}) \Rightarrow \theta^*$
3. $\{(x_i^{test}, y_i^{test})\}_{i=1}^{m'} \sim Q_{X,Y}$
4. $\epsilon_{test} = \frac{1}{m} \sum_{i=1}^{m'} L(f_{\theta^*}(x_i^{test}), y_i^{test})$

Key assumption is that both the training and test set come from the same distribution.

Is it a realistic assumption?

Source domain and Target domain

In practice, the training distribution and the test distribution are often not the same.

→ We train an image classifier on a database of photos taken with a professional camera, and want our classifier to work on pictures taken with any smartphone camera.

→ Training distribution ≠ Test distribution

→ $Q_{X,Y} \neq P_{X,Y}$

Domain adaptation definitions

Source domain, target domain, and domain shift

We introduce some terminology from the Domain Adaptation domain:

- **Source domain $\boldsymbol{Q_{X,Y}}$.** The data distribution on which the model is trained using labeled examples.
  → photos taken with a professional camera.
- **Target domain $\boldsymbol{P_{X,Y}}$.** A different, yet "related" distribution on which it is required to perform a similar task.
  → photos taken with a smartphone.
- **Domain shift.** It is the statistical difference between different domains.
  → statistical difference between $\boldsymbol{Q_{X,Y}}$ and $\boldsymbol{P_{X,Y}}$.

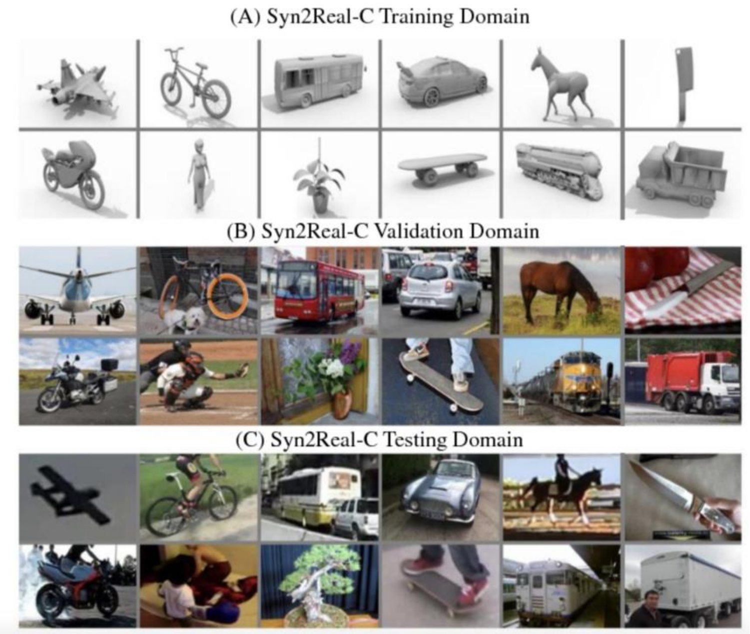
Domain adaptation definitions

Source domain, target domain, and domain shift

We introduce some terminology from the Domain Adaptation domain:

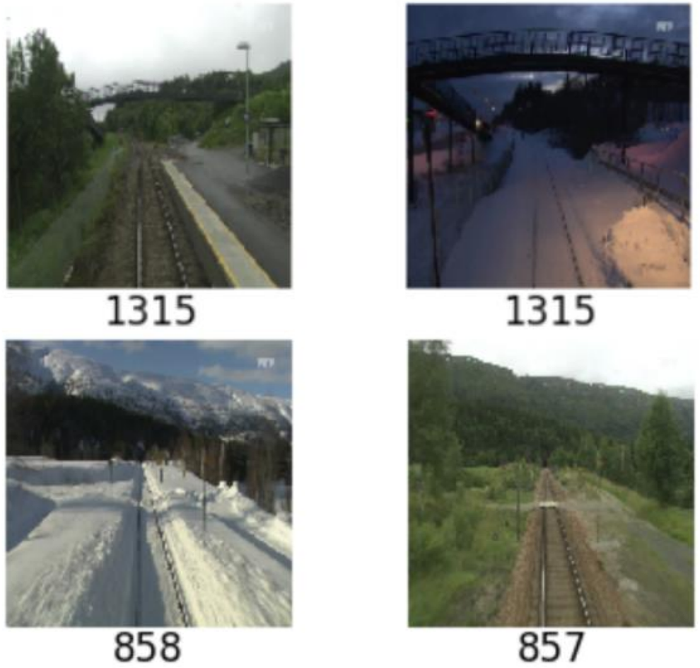
- **Source domain $\boldsymbol{Q_{X,Y}}$.** The data distribution on which the model is trained using labeled examples.
  → photos taken with a professional camera.
- **Target domain $\boldsymbol{P_{X,Y}}$.** A different, yet "related" distribution on which it is required to perform a similar task.
  → photos taken with a smartphone.
- **Domain shift.** It is the statistical difference between different domains.
  → statistical difference between $\boldsymbol{Q_{X,Y}}$ and $\boldsymbol{P_{X,Y}}$.

Domain adaptation examples



Train on synthetic samples and use with real samples.

Image from : Peng X. et al., “Syn2Real: A New Benchmark for Synthetic-to-Real Visual Domain Adaptation”



Same view from different seasons

Image from : Olid D. et al., “Single-View Place Recognition under Seasonal Changes”

Types of domain adaptation

Unsupervised domain adaptation

- Labeled samples for the source domain
  $Q_{X,Y} \sim \{(x_i^S, y_i^S)\}_{i=1}^{m_S} \coloneqq (X^S, Y^S)$
- Only unlabeled samples available for the target domain
  $P_X \sim \{x_i^T\}_{i=1}^{m_T} \coloneqq X^T$

Semi-supervised domain adaptation

- Labeled samples for the source domain
  $Q_{X,Y} \sim \{(x_i^S, y_i^S)\}_{i=1}^{m_S} \coloneqq (X^S, Y^S)$
- Unlabeled target samples + "Few" labeled target samples

Domain generalization

- Labeled samples for the multiple source domains
  $Q_{X,Y}^1 \sim \{(x_i^{S_1}, y_i^{S_1})\}_{i=1}^{m_{S_1}} \coloneqq (X^{S_1}, Y^{S_1})$
  $Q_{X,Y}^2 \sim \ldots$
  …
- **No** samples from the target domain available during training
- This problem is also called "out-of-distribution generalization"

Types of domain adaptation

Notice:

- We use both the samples from the source domain and from the target domain during training
- The target domain is different than what we use to call test set
- We need labelled samples from the target domain for testing, in all three scenarios

Domain adaptation

Unsupervised domain adaptation

Unsupervised domain adaptation

Let's assume, for simplicity and without loss of generalization, that $m_S = m_T = m$, i.e.,

- Source domain: $(X^S, Y^S) = \{(x_i^S, y_i^S)\}_{i=1}^{m} \sim Q_{X,Y}$
- Target domain: $X^T = \{x_i^T\}_{i=1}^{m} \sim P_X$

The goal in unsupervised domain adaptation is that of, given a hypothesis class H, to pick a function $h \in H$ such that

$$\epsilon_T(h) = \mathbb{E}[L(h(x), y)]$$

Is minimized, with $(x, y) \sim P_{X,Y}$.

Unsupervised domain adaptation

Assumptions

Machine Learning Data Analytics

1. **Covariate shifts.** P and Q satisfy the covariate shift assumption if the conditional label distribution does not change between source and target distribution.

$$\forall x \in X, y \in \{0, 1\} \Rightarrow P(y\,|x) = Q(y\,|\,\mathrm{x})$$

2. **Similarity of distributions.** Source and target (marginal) distribution should be similar.

$$Q_X \ldots <=> \ \ldots P_X$$

3. **Small joint error.** If I "had" labeled samples, the joint error should be small.

$$\epsilon_{joint} = \min\left[\frac{1}{m}\sum_{i=1}^{m} L\left(h\left(x_i^S\right), y^S\right) + \frac{1}{m}\sum_{i=1}^{m} L\left(h\left(x_i^T\right), y^T\right)\right] \approx 0$$

Domain Adaptation "Main Result"

The following "main result" has inspired many practical methods in domain adaptation.

Main result. H is a hypothesis class with $VC(H) = d$. We are given unlabeled samples from the target $P_X^{(m)}$ and labeled samples from the sources $Q_{X,Y}^{(m)}$. With probability $1 - \delta$, for any $h \in H$,

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{H\Delta H}\left(Q_X^{(m)}, P_X^{(m)}\right) + \epsilon_{joint}$$

(Target error $\leq$ source error + H-divergence + joint error)

Notice: a formal definition of the H-divergence is included in the extra slides, at the end of this presentation

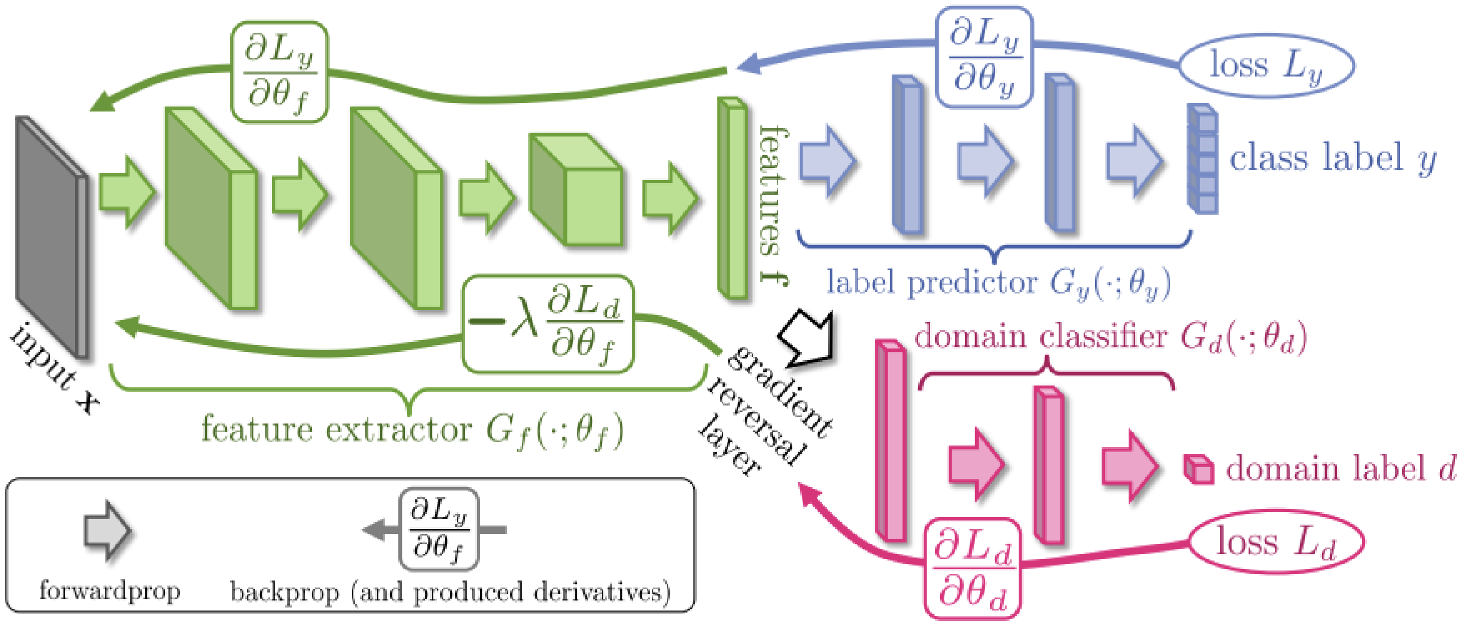
Practical Domain Adaptation methods

The main result resulted in many practical methods (approximation methods) in order to use the concept of divergence in the training itself.

- Classical domain adaptation methods
  - Metric learning
  - Sample re-weighting
  - Subspace alignment
  - …

- Deep Learning-based methods
  - Nowadays an hot topic of research

Domain adaptation: Ganin & Lempitsky method



- In general we want to learn a mapping (input embedding) such that performance on the task are maximised, but penalises the domain classification.

Image from: Ganin & Lempitsky, “Unsupervised Domain Adaptation by Backpropagation”

Domain adaptation

Domain generalization (OOD generalization)

Domain generalization
(also called, Out-of-distribution (OOD) generalization)

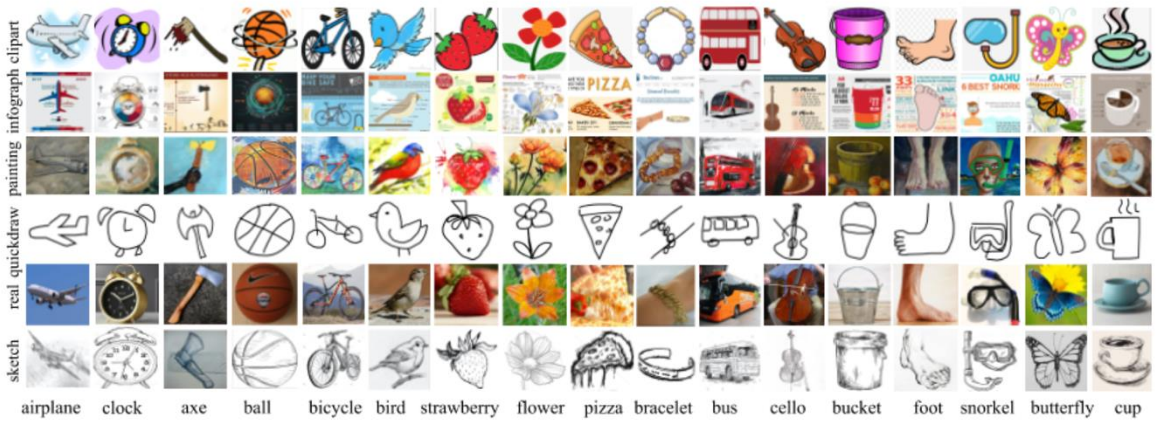
The problem of domain generalization (also called, out-of-distribution (OOD) generalization) can be formalized as follows:

- Training: $K = |E|$ training domains
  - $P^{(e)} \sim \{(x_i^e, y_i^e)\}_{i=1}^{m_e}$
  - $1 \le e \le |E|$
- Goal: find $h \in H$ that performs well in an unseen domain $|E| + 1$
  - $P^{(K+1)} \sim \left\{\left(x_i^{(K+1)}, y_i^{(K+1)}\right)\right\}_{i=1}^{m_{(K+1)}}$
- Minimize the risk in the new environment
  - $R^{(K+1)}(h) = \mathbb{E}_{(x,y)\sim P^{(k+1)}}\left[L(h(x), y)\right]$

Note: also in this setup, different environments need to be “related” to each other.

Example datasets for domain generalization

DomainNet



- http://ai.bu.edu/M3SDA/
- 345 classes
- Domains: clipart, real, sketch, infograph, paintings, drawings

PACS

- https://paperswithcode.com/dataset/pacs
- 7 categories
- Domains: photo, paintings, cartoon, sketch

Method 1: Baseline method.

We call “Baseline” method the approach that consists simply on minimizing the error on the available domains.

- **Training:** $\min_{f} \frac{1}{K} \sum_{j=1}^{k} \mathbb{E}_{(x,y)\sim P^{(j)}} [L(f(x), y)]$
- **Test:** $\mathbb{E}_{(x,y)\sim P^{(K+1)}} [L(f(x), y)]$
- “Do nothing” method

Method 2: Invariant representation.

Learn a representation that is invariant across different domains

- Use domain adversarial neural networks (DANN)
  - $\boldsymbol{\phi}$ (feature extraction)
  - $\boldsymbol{\omega} \circ \boldsymbol{\phi}$ (label classification)
  - $\boldsymbol{c} \circ \boldsymbol{\phi}$ (domain classification)
  - $loss = \frac{1}{K}\sum_{j=1}^{K} L(\boldsymbol{\omega} \circ \boldsymbol{\phi}(x), y) - \lambda \frac{1}{K}\sum_{j=1}^{K} L(\boldsymbol{c} \circ \boldsymbol{\phi}(x), y)$
  - $\min_{\phi,\omega} loss \quad \& \quad max_c \; loss$
- "Do something" method

Domain adaptation

Recap

Recap

- **Domain adaptation**
  - **Unsupervised domain adaptation**
    - Main result
    - Practical methods
  - **Semi-supervised domain adaptation**
  - **Domain generalization**
    - Baseline method
    - Invariant representations method

Extra slides

H-divergence

H-divergence is defined as:

$$2 \sup_{h \in H} \left| p_{x \in Q_X}(h(x) = 1) - p_{x \in P_X}(h(x) = 1) \right| \triangleq d_H(Q_X, P_X)$$

Lemma. The H-divergence $d_H(Q_X, P_X)$ can be estimated by $m_S = m_T = m$ samples from source and target domains, $VC(H) = d$, with probability $1 - \delta$,

$$d_H(Q_X, P_X) \leq d_H\left(Q_X^{(m)}, P_X^{(m)}\right) + 4\sqrt{\frac{d \log(2m) - \log(\frac{2}{5})}{m}}$$

Estimate H-divergence

Methods for estimating the H-divergence

The H-divergence can be computed by finding a classifier to separate source domain from target domain.

- Label all source samples as +1
- Label all target samples as 0
- Train a classifier to minimize the classification error:

$$\epsilon_{class} = \min_{h \in H} \left[\frac{1}{m} \sum_{i=1}^{m} 1\left(h\left(x_i^S\right) = 0\right) + \frac{1}{m} \sum_{i=1}^{m} 1\left(h\left(x_i^T\right) = 1\right) \right]$$

The classification loss in inversely proportional to the H-divergence,

$$\frac{1}{2} d_H \left(Q_X^{(m)}, P_X^{(m)} \right) = 1 - \epsilon_{class}$$

Symmetric difference hypothesis space

Definition

Definition. For the hypothesis class H, the symmetric difference hypothesis space $H\Delta H$ is the set of disagreements between any two hypothesis in H.

$$H\Delta H = \{g(x) = h(x) \oplus h'(x) | h, h' \in H\}$$