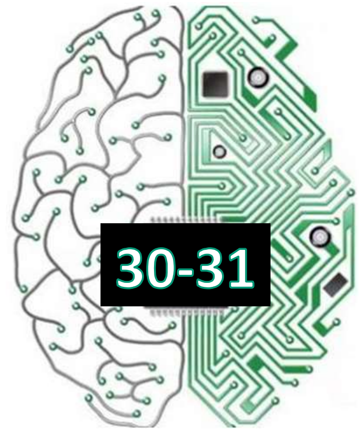
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#

Deep Learning

Unit-6: Representation Learning (Part III)
Unit-7: Structured Probabilistic Models (Part-I)



Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

Transfer Learning

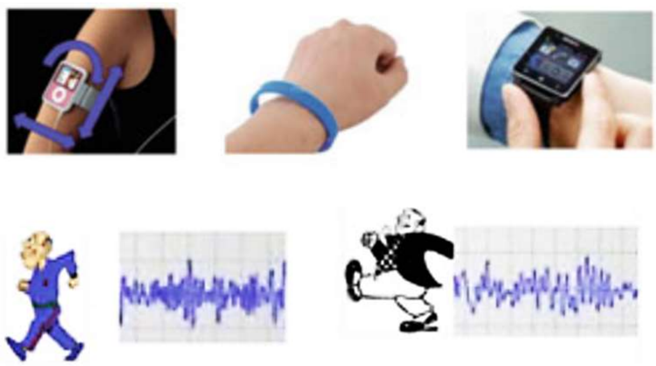
2

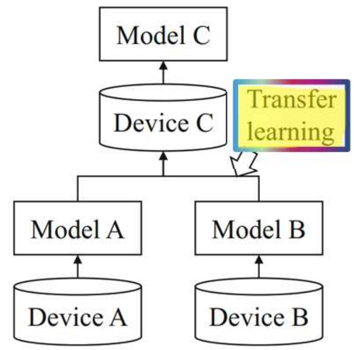
- Transfer learning aims to solve the new problem by leveraging the similarity of data (task or models) between the old problem and the new one to perform knowledge (experience, rules, etc.) transfer.
- As an important branch of machine learning, focuses on the process of leveraging the learned knowledge to facilitate the learning of new ability, which increases the effectiveness and efficiency.

Real-Life Example

3

- **Human Activity Recognition**





With the common knowledge from A and B, the model of C will be trained more efficiently

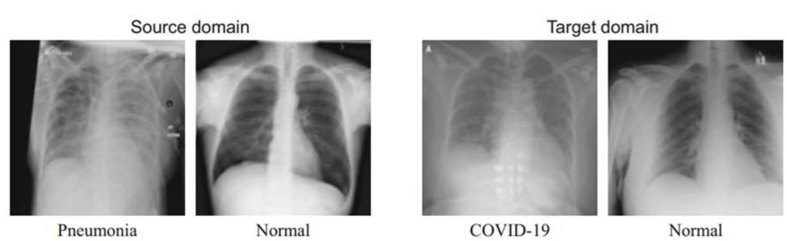
This will prevent re-training from the data of C

Applications of Transfer Learning

4

- Computer vision
- NLP
- Ubiquitous Computing
- Healthcare
- Speech Recognition
- Cross-lingual adaptation for few shot learning of resource-poor languages

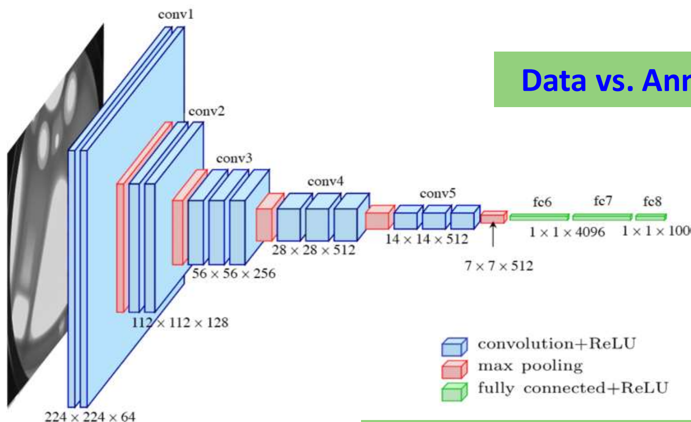
Image dataset 1 Image dataset 2



Why Transfer Learning?

5

- **How do you build a classifier that can be trained in a few minutes on a CPU with very little data?**



Data vs. Annotation

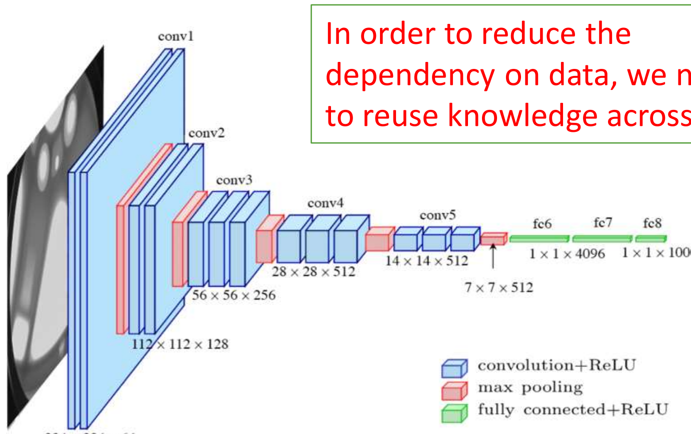
Limited Data vs. Generalization

Why Transfer Learning?

6

- **How do you build a classifier that can be trained in a few minutes on a CPU with very little data?**

In order to reduce the dependency on data, we need to reuse knowledge across tasks

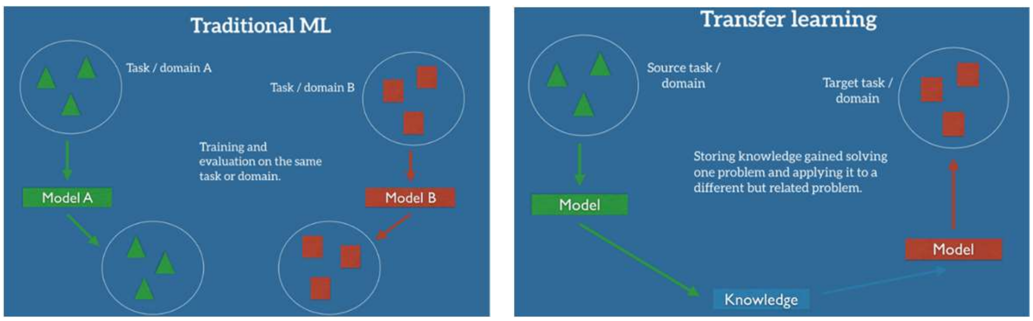


Data vs. Computation

Transfer Learning (TL) vs. Traditional ML/DL

7

- **Transfer learning (TL):** "focuses on storing knowledge gained while solving one problem and applying it to a different but related problem"
  – allows such knowledge transfer to take place even if the ***domain*** and ***tasks*** are different



Different with respect to three key aspects: **1) Data distribution 2) Data annotation 3) Model**

Formal Definition

8

- A **Domain** consists of two components: $D = \{\chi, P(X)\}$
  - Feature space: χ
  - Marginal Distribution: $P(X); X = \{x_1 \ \dots, x_n\}, x_i \in \chi$

- For a given Domain, a **Task** is defined by two components:

$$T = \{\mathcal{Y}, P(Y|X)\} = \{\mathcal{Y}, \eta\}; Y = \{y_1, \dots, y_n\}, y_i \in \mathcal{Y}$$

  – Label space: $\mathcal{Y}$
  – A predictive function η, learned from feature vector/label pairs $(x_i, y_i), x_i \in \chi, y_i \in \mathcal{Y}$. For each feature vector in the domain, η predicts its corresponding label $\eta(x_i) = y_i$.

Scenarios of TL

9

- **Different features spaces among source and target**

$$\mathcal{X}_{source} \neq \mathcal{X}_{target}$$

- **Different marginal probabilities among source and target**

$$P_s(X) \neq P_t(X)$$

- **Different labels among source and target**

$$\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$$

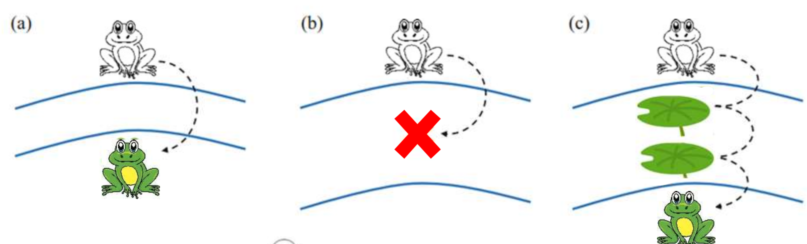
- **Different conditional probabilities distribution among source and target task**

$$P_s(Y|X) \neq P_t(Y|X)$$

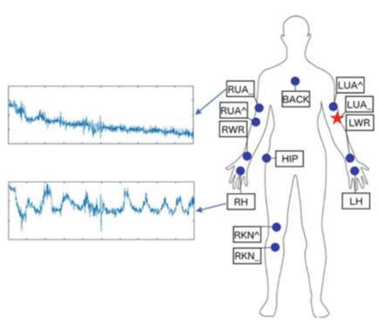
Fundamental Problems in Transfer Learning

10

- **When to transfer**
  - ➢ The core of transfer learning to find and exploit the ***similarity*** between two domains.



- **What/Where to transfer**
  - ➢ Selecting appropriate source domain
  - ➢ Selecting appropriate samples

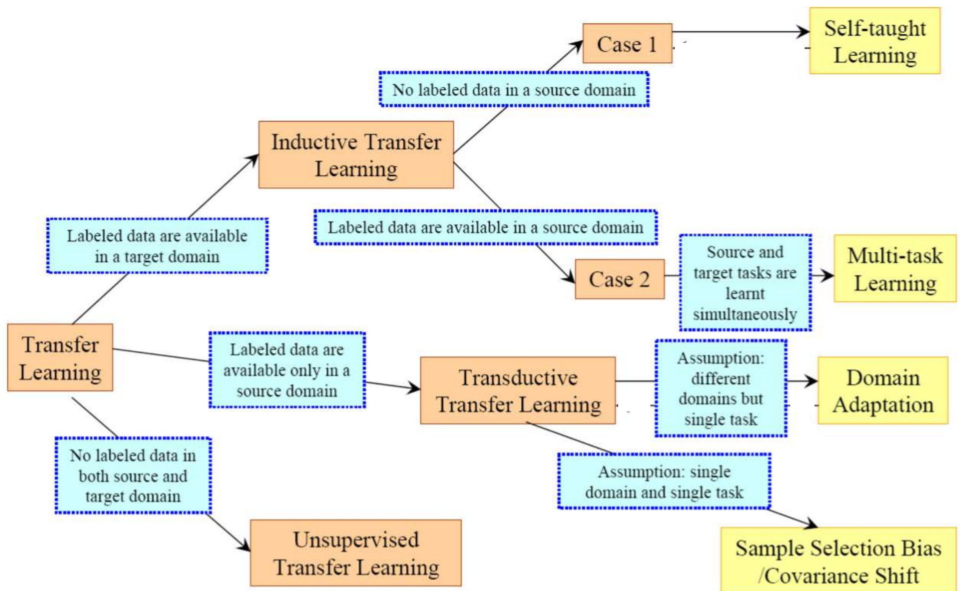


- **How to transfer**
  - ➢ Transfer learning strategies

Negative Transfer Learning: the knowledge borrowed from the source domain has negative effects on the target domain.

Transfer Learning Settings

11



Approaches to Transfer Learning

12

- Instance Transfer
- Feature Representation Transfer
- Parameter Transfer
- Relational Knowledge Transfer

Transfer Learning for Deep Learning

13

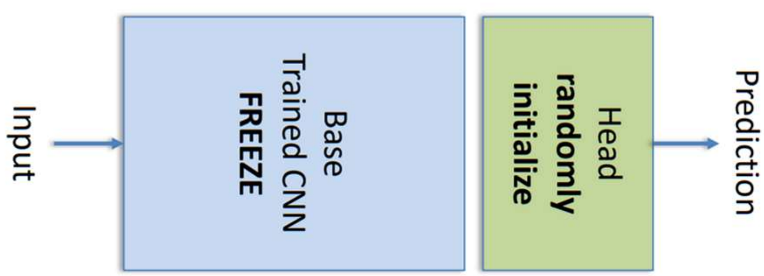
- **What people think**
  - you can't do deep learning unless you have a million labeled examples.

- **What people can do, instead**
  - You can learn representations from unlabeled data
  - You can train on a nearby objective for which is easy to generate labels (imageNet).
  - You can transfer learned representations from a related task.

Representation Extraction

14

- Use representations learned by big net to extract features from new samples, which are then fed to a new classifier

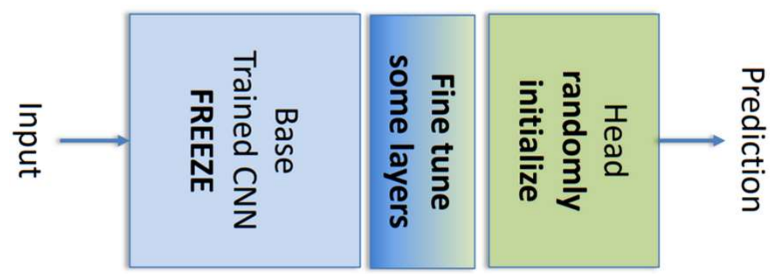


'avlos Protopapas

Fine-tuning

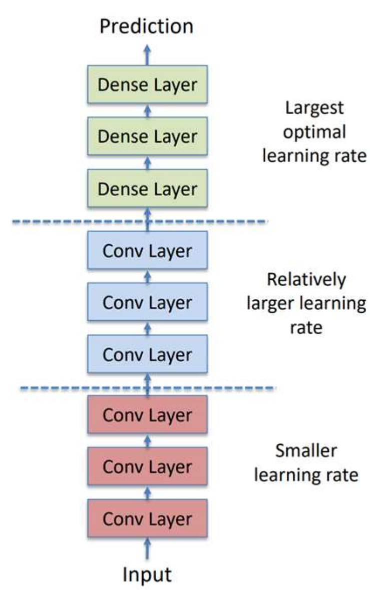
15

- Up to now we have frozen the entire convolutional base.



Differential Learning Rates

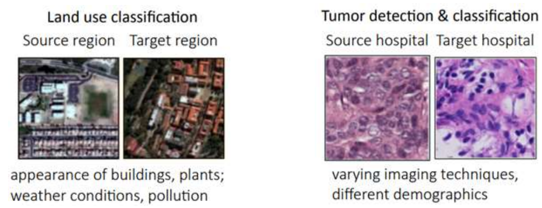
16



Domain Adaptation

17

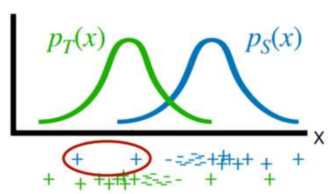
- A form of transfer learning, with access to unlabeled target domain data during training
- A kind of transductive transfer learning
- Common assumptions:
  - $p_S(y \mid x) = p_T(y \mid x)$
  - There exists a single hypothesis $f(y|x)$ with low error.
- **Example:**



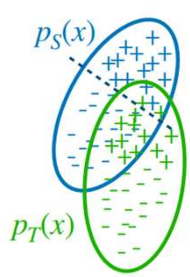
Domain Adaptation

18

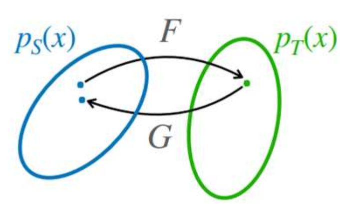
- **Algorithms**
  - Data reweighting



  - Feature Alignment

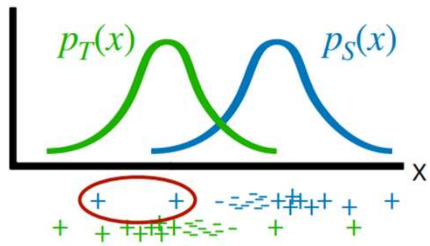


  - Domain Translation



Domain Adaptation Problem

19

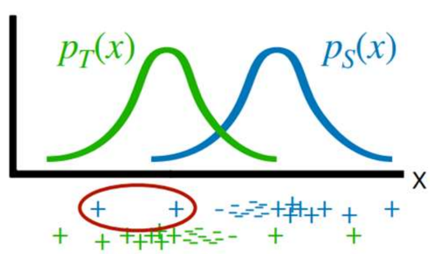


Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

How can we learn a classifier that does well on $p_T(x)$?
(using labeled data from $p_S(x)$ & unlabeled data from $p_T(x)$)

Domain Adaptation Problem

20



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

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

Why does this make sense mathematically?

Domain adaptation via importance sampling

21

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

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

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

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

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

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

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

Domain adaptation via importance sampling

22

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

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

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

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

Bayes rule:

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

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

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

a constant

can estimate with binary classifier!

Domain adaptation via importance sampling

23

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

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

a constant

can estimate with binary classifier!

Full algorithm:

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

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Domain adaptation via importance sampling

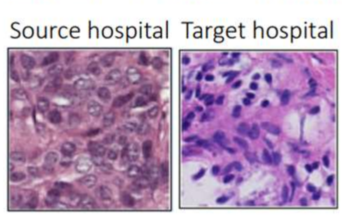
24

- **Drawback:**

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

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

Tumor detection & classification

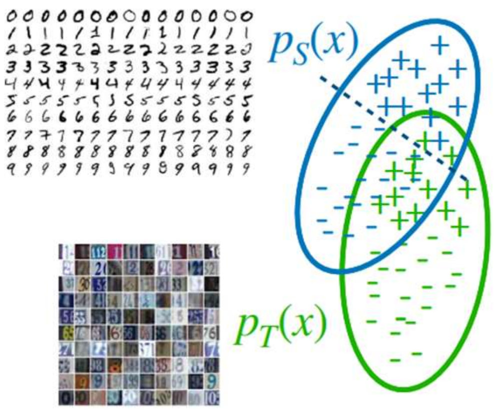


—> Source probably won't cover target distr!

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Domain adaptation via feature alignment

25

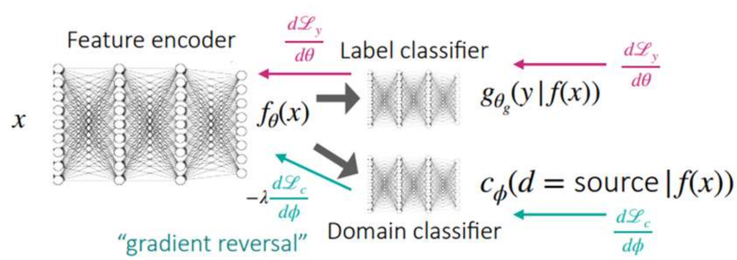


Can we align the features?

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

Domain adaptation via feature alignment

26



Full algorithm:

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

Doesn't require source data coverage!

Domain adaptation via feature alignment

27

- **Drawbacks**
  - Involves adversarial optimization
  - It may be hard to align features

    Idea: translate between domains

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

Domain Translation with CycleGAN

28

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

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

Challenge: The mapping is underconstrained, can be arbitrary.
Can we encourage models to learn a consistent, bijective mapping?

Step 2: Train F and G to be cyclically consistent.
$F(G(x)) \approx x$ and $G(F(x)) \approx x$

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

Domain Translation with CycleGAN

29

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

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

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

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

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

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

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

30

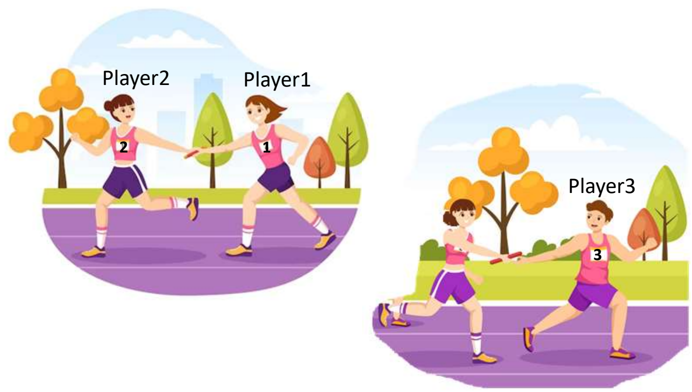
Unit-7: Structured Probabilistic Models for Deep Learning

Challenge of Unstructured Modelling

- Modeling a distribution over a random vector x containing n discrete variables capable of taking on k values each
- **Naive approach:** storing a ***lookup table*** with one probability value per possible outcome
  - Expensive!

The probability distributions encountered in real tasks are much simpler.

Most variables influence each other only indirectly.



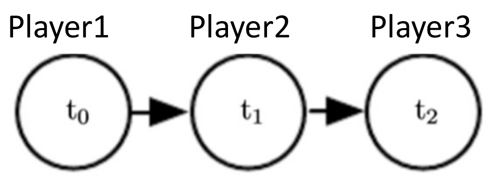
Structured Probabilistic Model

- **Structured probabilistic model**
  – a way of describing a probability distribution, ***using a graph*** (consisting of nodes and edges)
  – describes which random variables in the probability distribution interact with each other ***directly***.
  – allows the models to have ***significantly fewer parameters*** and therefore be estimated reliably from less data.

Directed Models

33

- **Belief network or Bayesian network:** Directed acyclic graph
- An arrow from a to b: distribution over b depends on the value of a



$$p(t_0, t_1, t_2) = p(t_0)p(t_1|t_0)p(t_2|t_1)$$

Dramatic savings in cost!

For a directed acyclic graph G: $P(\mathbf{x}) = \prod_i p(x_i | Pa_G(x_i))$

the set of parents of x_i in G

34

Questions?