When the **learning** distribution differs from the **target** (true) distribution

Imbalanced data sets

Learning from **positive examples** only

Semi-supervised learning

Active Learning

Domain **adaptation**

Antoine Cornuéjols

AgroParisTech – INRAE   MIA Paris-Saclay

EKINOCS research group

When $P_X(\text{train}) \neq P_X(\text{test})$

$P_X(\text{train}) \neq P_X(\text{test})$

- In which scenarios?

$P_X(\text{train}) \neq P_X(\text{test})$

In which scenarios?

1. Classes are severely **unbalanced**
2. Learning from **positive** examples **only**
3. **Semi-supervised** learning
4. **Active** learning

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

Illustrations

- Rare pathologies
- Anomaly detection
- Fraud
- Rare species
  - E.g. Pl@ntNet: **46,000** species, but only ~**1000** well represented

Remedies

Remedies

- If **enough** data
  - **undersample** the over-represented classes

Remedies

- If **enough** data
  - **undersample** the over-represented classes
- If **not enough** data

Remedies

- If **enough** data
  - **undersample** the over-represented classes
- If **not enough** data
  - **oversample** the under-represented classes
    - Create **noisy** clones of the data points
    - Create **new** data points generated by **well chosen transformations**
      - E.g. respecting **invariances** (E.g. translations, rotations, change of luminosity, …)

Remedies

- If **enough** data
  - **undersample** the over-represented classes

- If **not enough** data
  - **oversample** the under-represented classes
    - Create **noisy** clones of the data points
    - Create **new** data points generated by **well chosen transformations**
      - E.g. respecting **invariances** (E.g. translations, rotations, change of luminosity, …)

- Modify the **loss function**
  - **Penalize** more the errors on the under-represented class

$$\ell_{\hat{\mathrm{M}},\mathrm{m}} P_{\hat{\mathrm{M}},\mathrm{m}} + \ell_{\hat{\mathrm{m}},\mathrm{M}} P_{\hat{\mathrm{m}},\mathrm{M}} \quad \text{with} \quad \ell_{\hat{\mathrm{M}},\mathrm{m}} >> \ell_{\hat{\mathrm{m}},\mathrm{M}}$$

Proportion of all points where points of the minority class are misclassified as from the Majority one

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

Scenarios for learning from positive examples only

- ???

Scenarios for learning from positive examples only

- Collaborative science
  - Biodiversity
  - E.g. Pl@ntNet
    - The users take pictures of plants: positive examples
    - That does not say: “these other plants were not present”

- Medicine
  - Reports of subjects with some disease does not say how many and which ones do not have the disease

- Adds on web pages
  - Pages that have **not been visited** are not necessarily uninteresting

Scenarios for learning from positive examples only

- In general
  - Detecting absence can be more difficult than detecting presence

Possibly **lots** of **false negative**

The fully observable case

- We look for a **hypothesis** $h : \mathcal{X} \to [0,1]^L$ A **vector** of predictions
  where *L* is the number of possible classes (labels)

- We want to **minimize the risk** $R(h) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})}\ \ell(h(\mathbf{x}), \mathbf{y})$
  with *loss function* $\ell : [0,1]^L \times \mathcal{Y} \to \mathbb{R}$
  (e.g. binary cross-entropy)

$$\ell_{\mathrm{BCE}}(h(\mathbf{x}_n), \mathbf{y}_n) = -\frac{1}{L}\sum_{i=1}^{L} P(\mathbf{y}_n^i = 1|\mathbf{x}_n)\log\big(h(\mathbf{x}_n^i)\big) + P(\mathbf{y}_n^i = 0|\mathbf{x}_n)\log\big(1 - h(\mathbf{x}_n^i)\big)$$

- Given a dataset $\mathcal{S} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{1\le n\le N}$
  we want to find a hypothesis that minimizes the empirical risk

$$\hat{h}_{\text{fully}} = \operatorname*{ArgMin}_{h\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N} \ell\big(h(\mathbf{x}_n), \mathbf{y}_n\big)$$

The partially observable case

- We look for a **hypothesis** $h_{\text{partial}} : \mathcal{X} \to [0,1]^L$

- During training, we observe $\mathbf{z}_n \in \mathcal{Z} = \{0, 1, \oslash\}^L$

  where $\mathbf{z}_n^i = \oslash$ ← indicates that the i^{th} label is unobserved

  and only one $\mathbf{z}_n^i = 1$

- Given a dataset $\mathcal{S} = \{(\mathbf{x}_n, \mathbf{z}_n)\}_{1 \le n \le N}$

  we want to find a hypothesis that

  minimizes the empirical risk

$$\hat{h}_{\text{partial}} = \underset{h \in \mathcal{H}}{\operatorname{ArgMin}} \frac{1}{N} \sum_{n=1}^{N} \ell\big(h(\mathbf{x}_n), \mathbf{z}_n\big)$$

Approach “assume **unobserved** are **negative**”

- Assume that all **unobserved** labels are **negative**

$$P(\mathbf{y}_n^i = 1 | \mathbf{x}_n) = 0 \quad \text{if} \;\; \mathbf{z}_n^i = \varnothing$$

- The resulting loss is

$$\ell_{\mathrm{AN}}(h(\mathbf{x}_n), \mathbf{y}_n) = -\frac{1}{L} \sum_{i=1}^{L} \mathbb{1}_{[\mathbf{z}_n^i = 1]} \log\big(h(\mathbf{x}_n^i)\big) + \mathbb{1}_{[\mathbf{z}_n^i \neq 1]} \log\big(1 - h(\mathbf{x}_n^i)\big)$$

$$\mathbb{1}_{[\mathbf{z}_n^i = 1]} = 1 \quad \text{if } \mathbf{z}_n^i = 1 \quad \text{and 0, otherwise}$$

- We expect **false negatives**

Approach “assume **unobserved** are **negative**” + **smoothing**

- Assume that all **unobserved** labels are **negative**

$$P(\mathbf{y}_n^i = 1 | \mathbf{x}_n) = 0 \quad \text{if} \;\; \mathbf{z}_n^i = \varnothing$$

- And give **more weight to the observed examples**. The resulting loss is

$$\ell_{\text{AN-LS}}(h(\mathbf{x}_n), \mathbf{y}_n) = -\frac{1}{L} \sum_{i=1}^{L} \mathbb{1}_{[\mathbf{z}_n^i = 1]}^{0.95} \log\big(h(\mathbf{x}_n^i)\big) + \mathbb{1}_{[\mathbf{z}_n^i \neq 1]}^{0.05} \log\big(1 - h(\mathbf{x}_n^i)\big)$$

Observed as **positive**

No observation reported
Hence assumed as **negative**

Intuitively $R(\hat{h}_{\text{fully}}) \leq R(\hat{h}_{\text{partial}})$

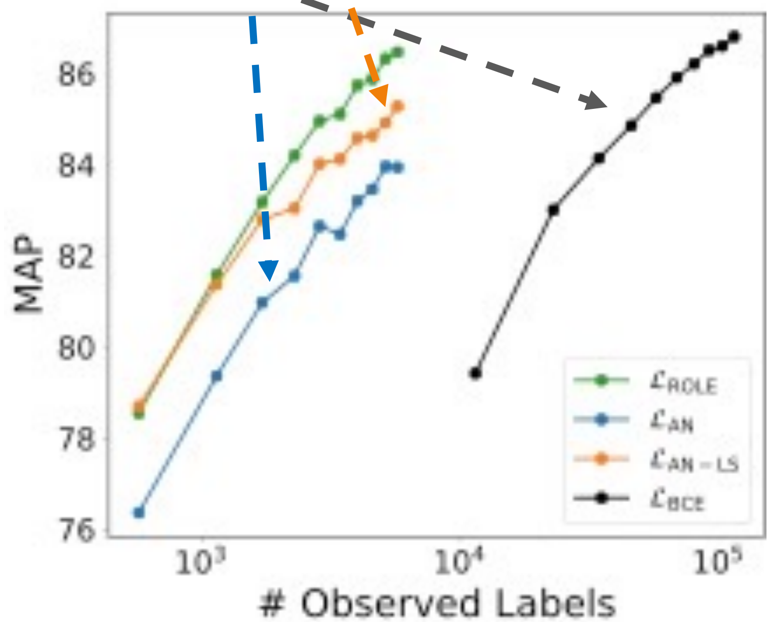
- But **by how much**?

- In the case of “assume unobserved = negative”

Intuitively $R(\hat{h}_{\text{fully}}) \leq R(\hat{h}_{\text{partial}})$

- But **by how much**?

- In the case of “assume unobserved = negative”



With 20 times fewer labeled examples, the performance is not that bad *on this dataset* compared to the fully observable case

COLE, Elijah, MAC AODHA, Oisin, LORIEUL, Titouan, *et al.* Multi-label learning from single positive labels. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021. p. 933-942.

Lessons

1. **Fomalize** the assumptions about your problem
   - The labelling process
   - The type of target (and hypothesis) function

2. Design a **loss function** appropriate for the problem
   - Able to **explore efficiently** the hypothesis space
     and to find a good minimum of the empirical risk

3. Design a good **evaluation scheme**

Learning from positive examples only: lots of approaches

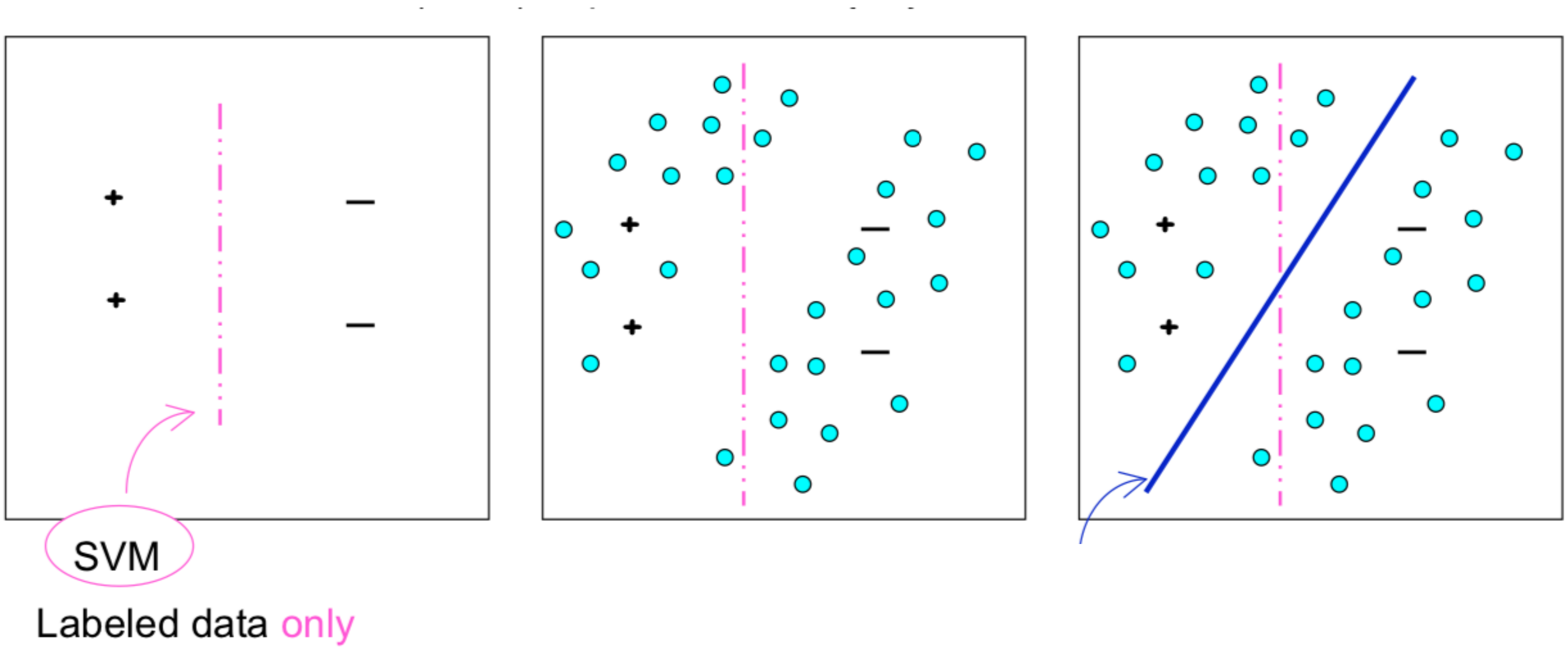
- Approaches
  - Assume that *the missing labels are negative*
  - *Ignore* the missing labels
  - Perform *label matrix reconstruction*
  - Learn *label correlations*
  - Learn *generative probabilistic models*
  - Train *label cleaning networks*

  - Related to **learning with label noise**
    - Here, some **unobserved labels** are incorrectly treated as being **absent**
  - Related to learning from a set of **positive examples**
    and a set of **unlabeled** ones (**PU** learning)

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

The idea



...

Semi-supervised learning

- **Unsupervised** learning $\mathbf{P}_{\mathcal{X}}$
- **Supervised** learning $\mathbf{P}_{\mathcal{Y}|\mathcal{X}}$

Semi-supervised learning

- **Unsupervised** learning $\mathbf{P}_{\mathcal{X}}$
- **Supervised** learning $\mathbf{P}_{\mathcal{Y}|\mathcal{X}}$

When can **unsupervised** learning help **supervised** learning?

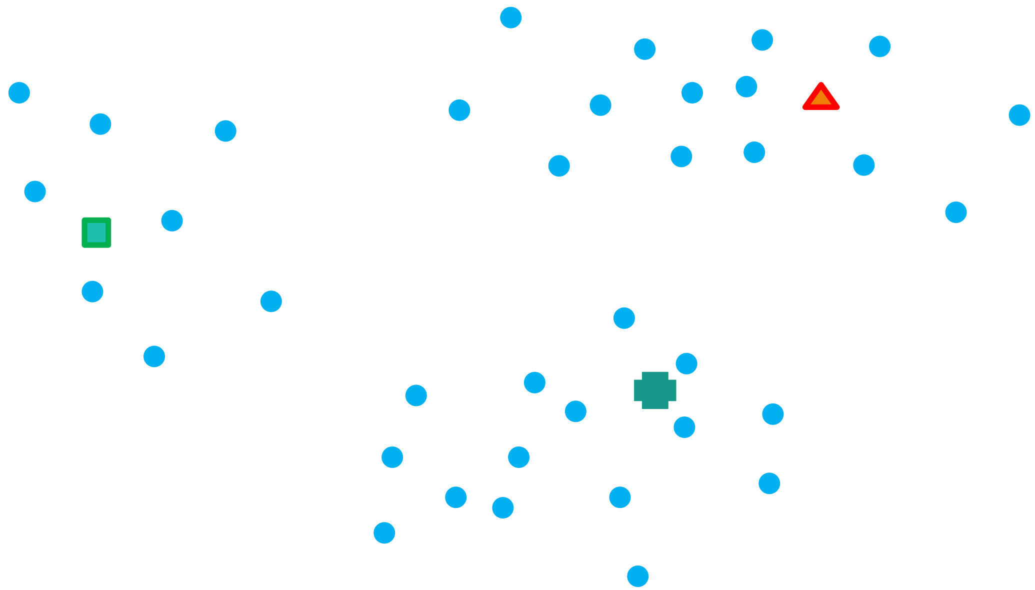
The underlying main idea:

The decision function (hypothesis h) **should not cut** through **high density** regions

Semi-supervised learning

Simplest approach

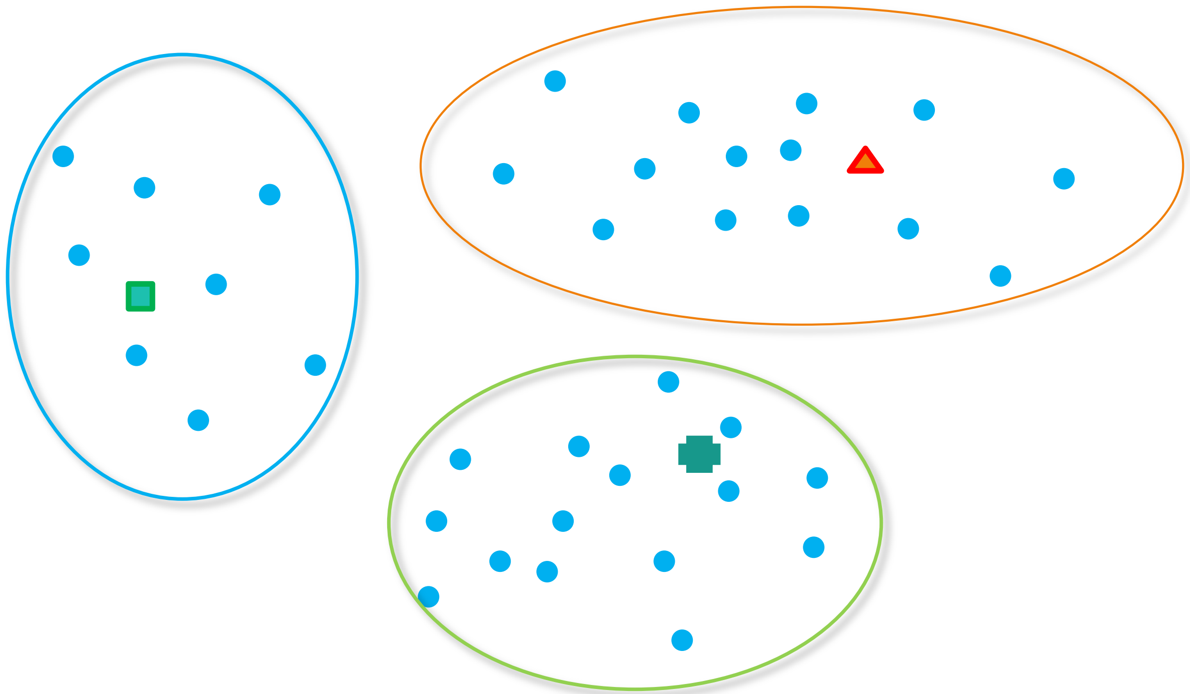
1. Compute a **clustering** of the all data (labeled and unlabeled)
2. For each cluster, **assign its class** to the majority vote of the labeled examples that belong to it



Semi-supervised learning

Simplest approach

1. Compute a **clustering** of the all data (labeled and unlabeled)
2. For each cluster, **assign its class** to the majority vote of the labeled examples that belong to it



Self-training approach

1. Given $\mathcal{S}_L = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq l}$ and $\mathcal{S}_U = \{(\mathbf{x}_j)\}_{1 \leq j \leq u}$
2. Train on S_L to obtain h_1
3. Apply h_1 to S_U
4. Remove a set of unlabeled data from S_U and add them to S_L (the one where $h(\mathbf{x})$ is the more confident) with the label $h(\mathbf{x})$
5. Go to 2 and **repeat** until **convergence**

Semi-supervised learning

- Idea: endow unlabeled data with pseudo-labels
  (the likeliest class at time *t*)

$$y_i = \begin{cases} 1 & \text{if } i = \text{argmax}_{i \in \{1,\ldots,C\}} \, h_i^t(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

Output of the i^th output neuron

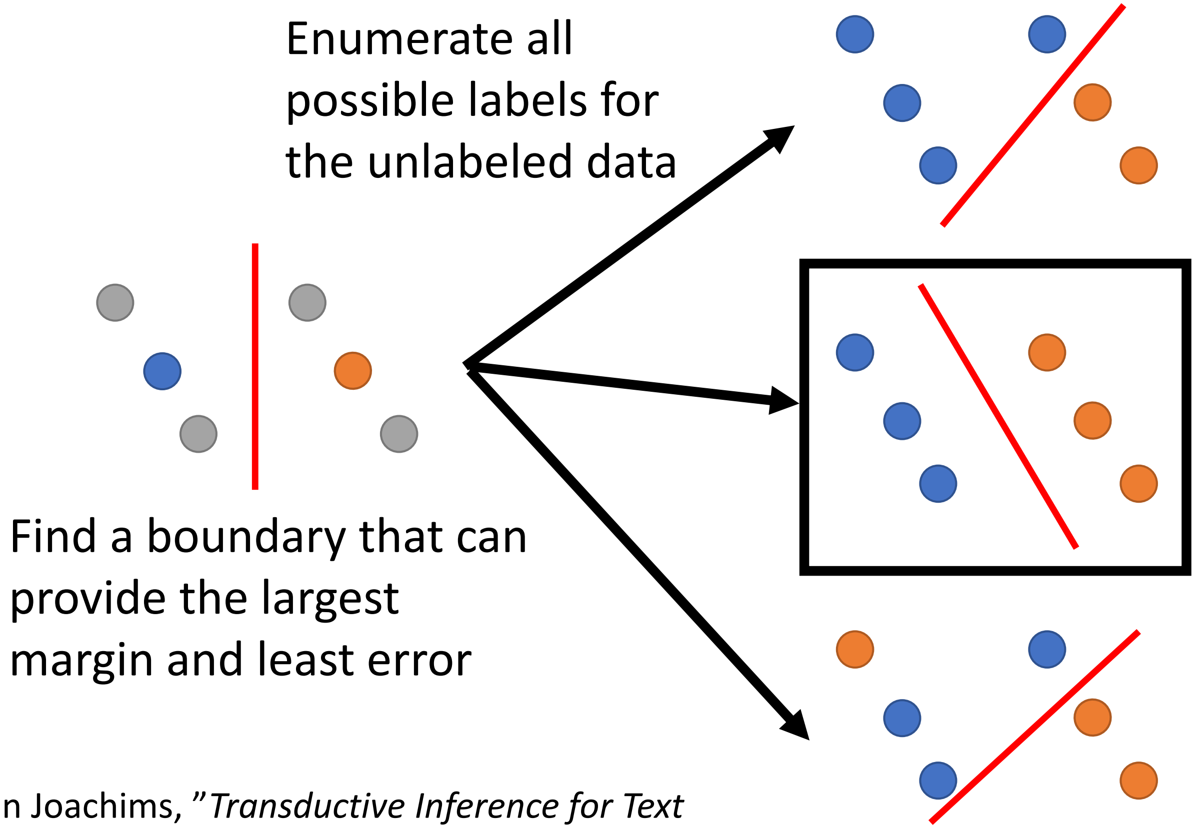
- Train with the empirical risk:

$$R_{\text{emp}}(h) = \frac{1}{m_l} \sum_{i=1}^{m_l} \sum_{j=1}^{C} \ell(h_j(\mathbf{x}_i), y_j^i) + \alpha(t) \frac{1}{m_u} \sum_{i=1}^{m_u} \sum_{j=1}^{C} \ell(h_j(\mathbf{x}_i), \underbrace{y_j^i}_{\text{pseudo-label}})$$

Crucial to set $\alpha(t)$ with great care

[Dong-Hyun Lee (2013) "*Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks*", ICML-2013]

Transductive SVM approach

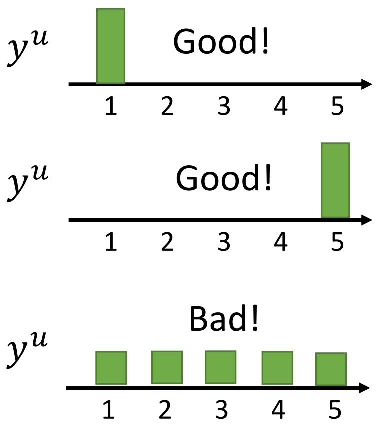


Thorsten Joachims, ”*Transductive Inference for Text Classification using Support Vector Machines”,* ICML, 1999

Semi-supervised learning

Entropy regularization approach

$$\hat{h} = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \Bigg[\underbrace{\frac{1}{l} \sum_{i=1}^{l} \ell(h(\mathbf{x}_i), y_i)}_{\text{Empirical risk on labeled data}} + \lambda \underbrace{\sum_{j=1}^{u} -h(\mathbf{x}_j) \log h(\mathbf{x}_j)}_{\text{Entropy of the predictions}} \Bigg]$$



$E(y^u) = 0$

$E(y^u) = 0$

$$E(y^u) = -ln\left(\frac{1}{5}\right) = ln5$$

- You have to make assumptions about what you think is reasonable as a bias
  - E.g. that classes are separated by low density regions

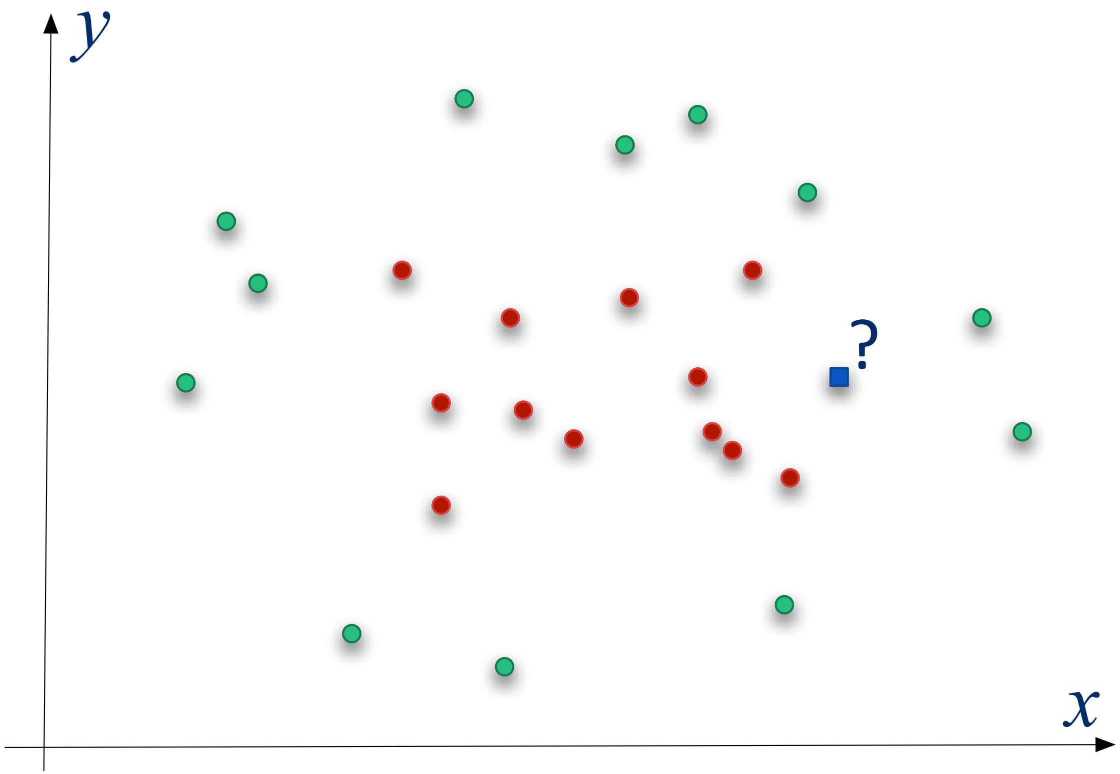
- Then, you show that if the assumption is met by Nature, then you find a correct hypothesis

A **remark** on semi-supervised learning

- Could be regarded as **transductive learning** where one wants to label unlabeled training instances

Transductive learning

- I know in advance where I will be queried



Transductive learning

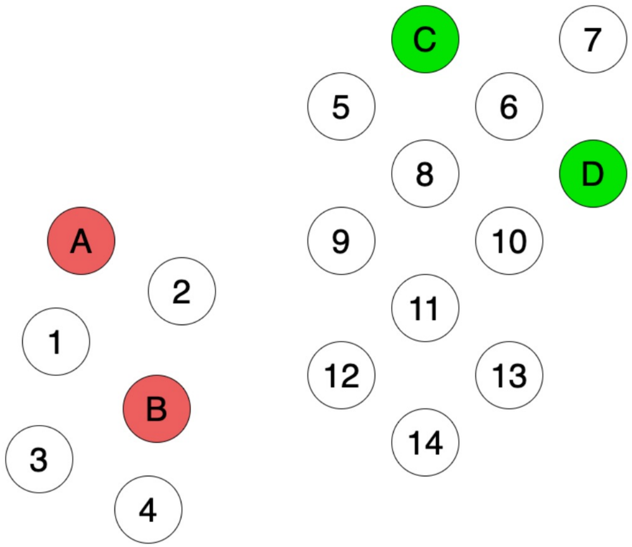
- "When solving a problem of interest, do not solve a more general problem as **an intermediate step**.

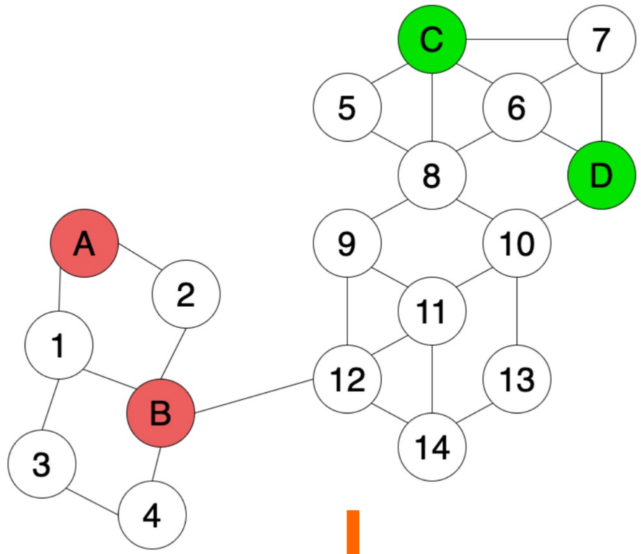
  Try to get the answer that you really need but not a more general one."

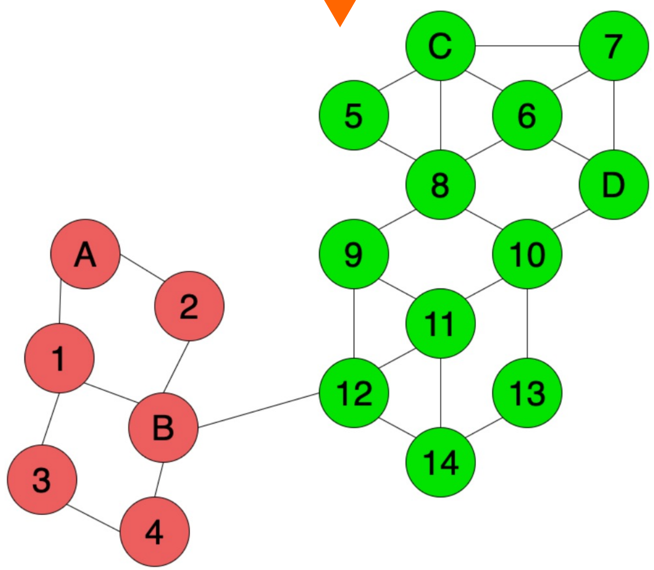
  (Vapnik, 1995)

Semi supervised learning with **transductive** learning

- Graph-Based labelling







Then **learn** a hypothesis on the new training set

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

Active learning

- When the learner can **actively ask** for pieces of information
  - Labels of selected **examples**
  - Values of some selected **descriptors**
    - E.g. ask for a medical examination

- Examples
  - MasterMind
  - Scientific activity

Active learning

- When the learner can **actively ask** for pieces of information
  - Labels of selected **examples**
  - Values of some selected **descriptors**
    - E.g. ask for a medical examination

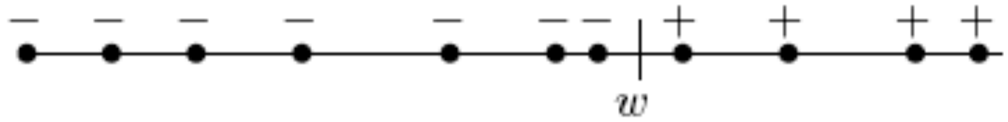
- The **hope**
  - Need of **less** (costly) examples
  - Having a **faster** convergence rate

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \boxed{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}} \right] \;>\; 1 - \delta$$

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \boxed{\sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2\,m}}} \right] \;>\; 1 - \delta$$

Active learning

$$h_w(x) = \begin{cases} 1 & \text{if } x \geq w \\ 0 & \text{if } x < w \end{cases}$$



How to find the **best** threshold from querying points?

- By **random** selection of points $m = \mathcal{O}(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$
- By **active** selection $m = \mathcal{O}(\log \frac{1}{\varepsilon})$

Much faster!

Active learning

- Two main approaches
  - “**Constructive**” approach
    - The learner **constructs** queries
  - ”**Selective**” (pool-based) approach
    - The learner **selects** points among the **unsupervised** ones

Why is the **constructive** approach sometimes **not** applicable?

How to **select** the examples? (some ideas)

- The more **informative** examples

1. The ones where the **confidence** of the current hypothesis is the **lowest**
   - Measured by a **probability**

$$\mathbf{x}^\star \;=\; \underset{\mathbf{x} \in \mathcal{S}_U}{\text{ArgMax}} \;\; \text{Uncertain}(\mathbf{x}) \qquad\qquad \text{Uncertain}(\mathbf{x}) \;=\; \frac{1}{\text{ArgMax}_{y \in \mathcal{Y}}\, p\big(h_t(\mathbf{x}) = y\big)}$$

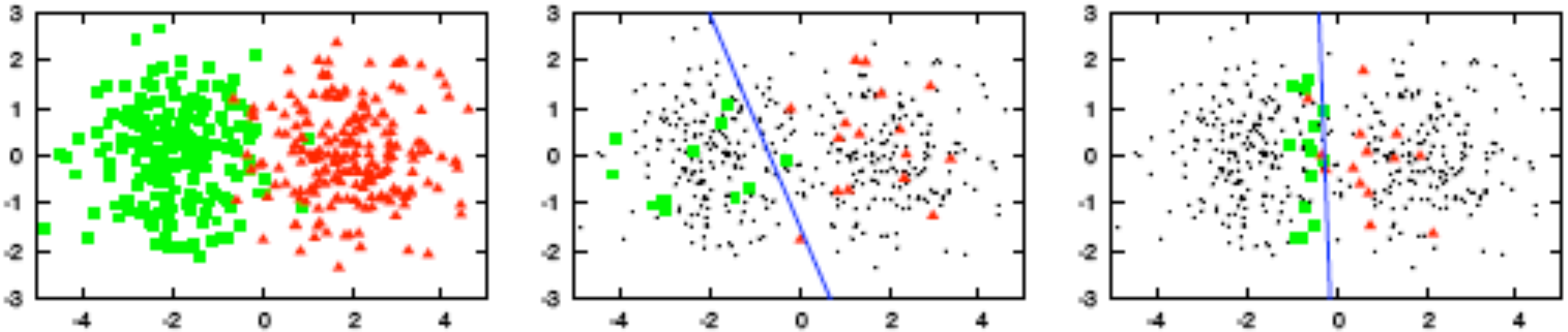
$$\mathbf{x}^\star \;=\; \underset{\mathbf{x} \in \mathcal{S}_U}{\text{ArgMax}} \left\{ -\sum_i p\big(h_t(\mathbf{x}) = y_i\big)\; \log p\big(h_t(\mathbf{x}) = y_i\big) \right\}$$

Entropy criyeria

   - Measured by **distance** to the decision function

2. Learn an **ensemble** of hypotheses and select the examples where they **disagree** the most

Illustration



(a) (b) (c)

Figure 2: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (accuracy = 0.7). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (accuracy = 0.9).

...

- What is the danger?

Active Learning

- What is the **danger**?

  - **No** more **theoretical** guarantees

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \;\leq\; R_{\text{Emp}}(h) \;+\; \boxed{\sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2\,m}}} \right] \;>\; 1 - \delta$$

Does not make sense anymore!!

  - Why?

Active learning: **lessons**

- Active learning is **not much used** in practice

    1. **Costly** to identify informative examples

    2. **Risk** of ignoring important regions of X

- Interesting: learning under **budget constraints**

    – What measurements should I made under some budget constraints?

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

Different types of transfers

- Domain **adaptation**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different distributions P_X
- Concept **shift**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different distributions $P_{Y|X}$
- **Transfer** learning
  - $X_S \neq X_T$ and/or $Y_S \neq Y_T$

Domain adaptation

- **Covariate** shift
  - We assume $X_S = X_T$ (same **input** space)

Training data

Test data

Source domain

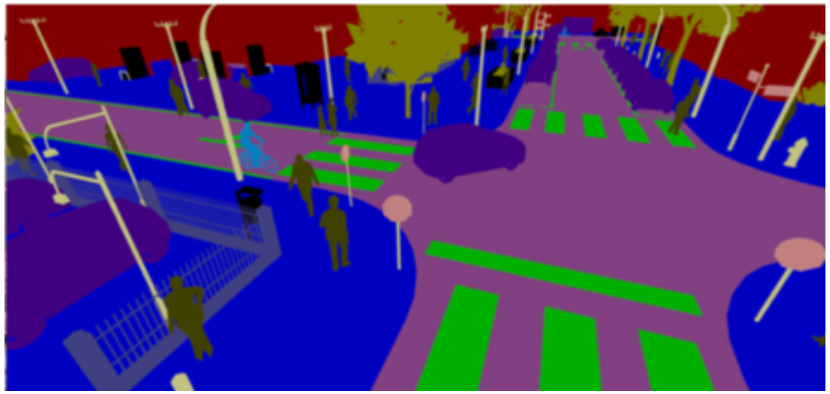
Target domain

- Covariate shift (suppose same input size and resolution)

$\mathbf{x}_S$

$\mathbf{x}_T$

y_S



$y_T = ?$

Source domain (simulated images)

Target domain (real images)

Concept shifts: illustrations

- **Spam** filtering
  - **Not** the same user: $P_{Y|X}$ may differ
    - E.g. for me conference announcements are important, but could be an annoyance to someone else

- **Changes in the tastes** or expectations of the consumers

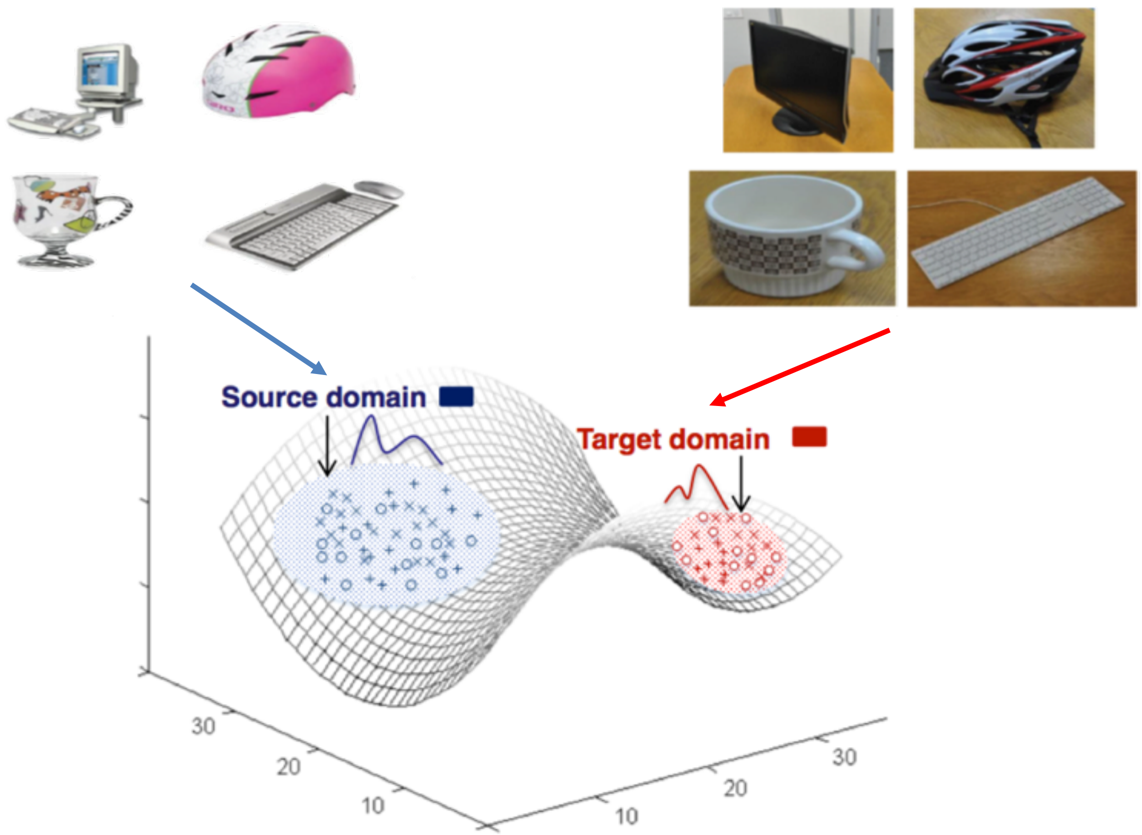
- **Changes** in medicine
  - E.g. the prevalence of flu differs from one season to another (P_X)
  - But this is still flu ($P_{Y|X}$)

Types of Domain Adaptation

- **Semi-supervised** DA (SSDA)
  - **Some labeled** target data, but not enough to train from it
  - Lots of unlabeled data
- **Unsupervised** DA (UDA)
  - **No labeled** target data
- **Source-free** DA (SFDA)
  - **No source data** (e.g. because of privacy concerns)
  - Only the **source hypothesis** h_S
  - And a few labeled target data

Covariate shift

- Difference in the P_X distribution between source and target domains: $\mathbf{P}_X^S \neq \mathbf{P}_X^T$



How to approach the problem

?

How to approach the problem

- **Very active** research area
  - Because of the **numerous** applications
- **Lots** of (heuristical) approaches

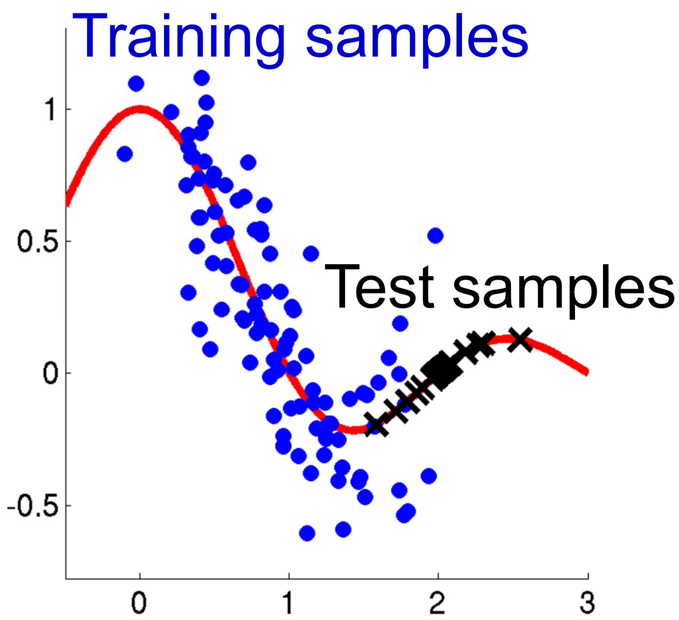
(Some) families of approaches

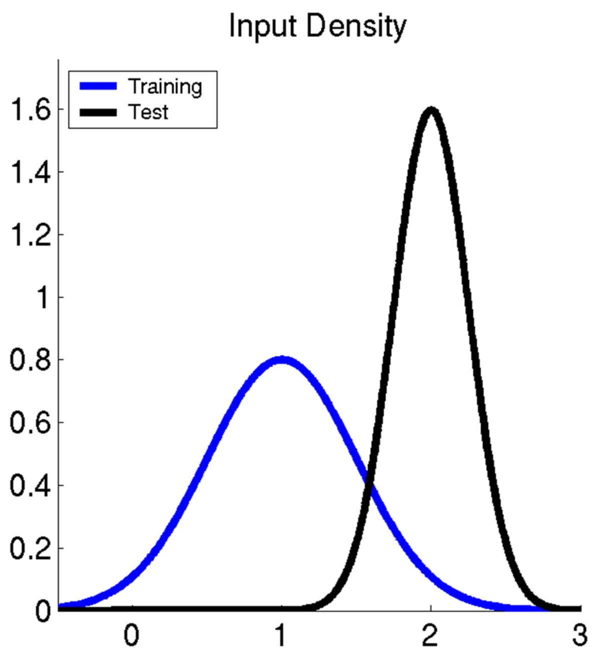
- **Change** the source distribution
  1. **Reweight** the source data
  2. Iteratively **self-label** the target data, and retrain

- Search for a **common description** subspace
  - Where the **source hypothesis works well** on the projected **source data**
  - And **hope** that it will work as well on the projected **target data**

DA by reweighting source data

- Here, a **regression** task





First analysis

$$R_{P_T}(h) = \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \sum_{(\mathbf{x}^t, y^t)} P_T(\mathbf{x}^t, y^t) \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \sum_{(\mathbf{x}^t,y^t)} P_T(\mathbf{x}^t, y^t) \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{P_T(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

Covariate shift [Shimodaira,'00]

$\Rightarrow$ Assume similar tasks, $P_S(y|\mathbf{x}) = P_T(y|\mathbf{x})$, then:

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{D_T(\mathbf{x}^t)P_T(y^t|\mathbf{x}^t)}{D_S(\mathbf{x}^t)P_S(y^t|\mathbf{x}^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t)\sim D_S} \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)} \mathop{\mathbf{E}}_{y^t\sim P_S(y^t|\mathbf{x}^t)} \}\mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$\Rightarrow$ **weighted error** on the source domain: $\omega(x^t) = \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)}$

Idea reweight labeled source data according to an estimate of $\omega(x^t)$:

$$\mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \omega(\mathbf{x}^t)\mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

Principle

- Law of large numbers
  - Sample averages converge to the population mean

$$\frac{1}{n}\sum_{i=1}^{n} A(x_i) \xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int A(x)\, \mathbf{p}_{train}(x)\, dx$$

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)} A(x_i) \xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int \frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)} A(x)\, \mathbf{p}_{train}(x)\, dx$$

$$\xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int A(x)\, \mathbf{p}_{test}(x)\, dx$$

  - But how to estimate $\boxed{\frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}}$?

Importance weighting

- A naïve estimation of $\boxed{\dfrac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}}$ does not work
  - Estimation density is too crude in high dimension space (and with few known testing instances)

- Idea of Sugiyama:
  - **Learn a parametric model of** $w(\mathbf{x}) = \dfrac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}$

$$\hat{w}(\mathbf{x}) = \sum_{j=1}^{J} \theta_j \, \phi_j(\mathbf{x})$$ and $$\hat{\mathbf{p}}_{test}(\mathbf{x}) = \hat{w}(\mathbf{x}) \, \mathbf{p}_{train}(\mathbf{x})$$

See [Sugiyama, Masashi, et al. "Direct importance estimation with model selection and its application to covariate shift adaptation." *Advances in neural information processing systems* 20 (2007)]

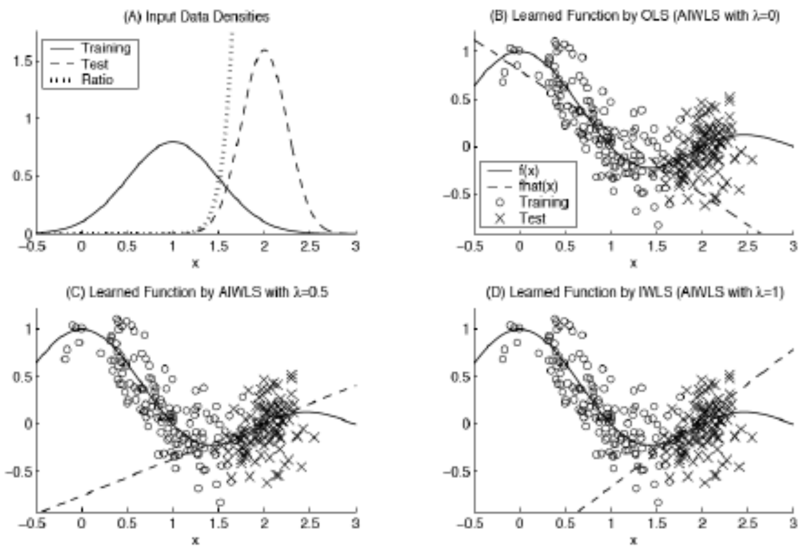
Covariate shift in regression

"Importance weighted" inductive criterion

Principle : weighting the classical ERM

$$R_{Cov}(h) \;=\; \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\mathbf{P}_{\mathcal{X}'}(\boldsymbol{x}_i)}{\mathbf{P}_{\mathcal{X}}(\boldsymbol{x}_i)}\right)^{\lambda}(h(\boldsymbol{x}_i - y_i)^2$$

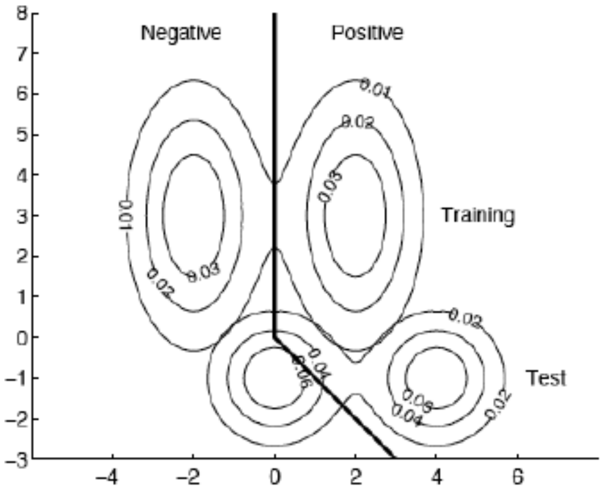
λ controls the stability / consistency (absence of bias)



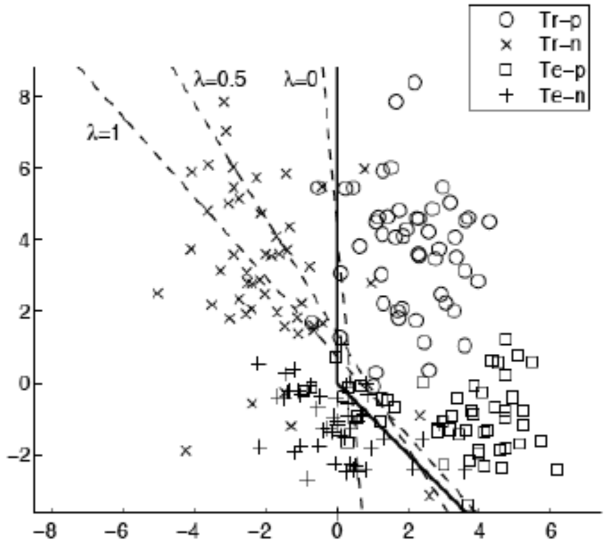
SKM07 M. Sugiyama and M. Kraudelat and K.-R. Müller (2007) *"Covariate Shift Adaptation by Importance Weighted Cross Validation"* Journal of Machine Learning Research, vol.8: 985-1005.

Covariate shift in classification

"Importance weighted" inductive criterion (classification task)



(a) Contours of training and test input densities.



(b) Optimal decision boundary (solid line) and learned boundaries (dashed lines). '○' and '×' denote the positive and negative training samples, while '□' and '+' denote the positive and negative test samples. Note that the test samples are not given in the training phase; they are plotted in the figure for illustration purposes.

SKM07 M. Sugiyama and M. Kraudelat and K.-R. Müller (2007) *"Covariate Shift Adaptation by Importance Weighted Cross Validation"* Journal of Machine Learning Research, vol.8: 985-1005.

The reweighting approach

$$\min_{\beta} \quad \left\| \frac{1}{n} \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_s^i) - \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{x}_t^i) \right\|^2$$

MMD

$$\text{s.t.} \quad \beta_i \in [0, B] \ , \ \forall\, 1 \leq i \leq n$$

Bound on the weights

$$\left| \sum_{i=1}^{n} \beta_i - n \right| \leq n\epsilon$$

Encourage the weights to define a probability distribution

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). **A kernel two-sample test**. *The Journal of Machine Learning Research*, *13*(1), 723-773.

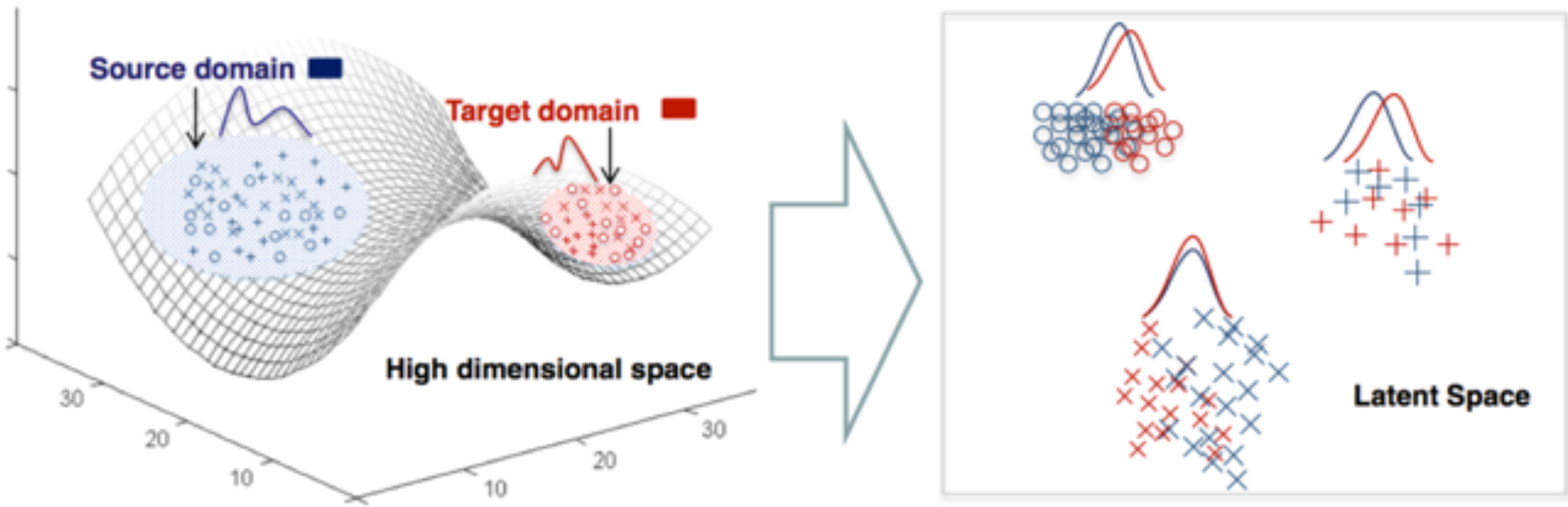
The reweighting approach

- … à la Fugiyama

  - **Complex** approach
  - **Not easy** to implement

Search for a **common description** space

- The idea



- The hope
  - If the **source hypothesis works well** on the projected **source data**
  - Then (?) it should/could work as well on the projected **target data**

Illustration by two algorithms

… among MANy others

1. Subspace alignment

2. Deep NNs

Subspace alignment algorithm

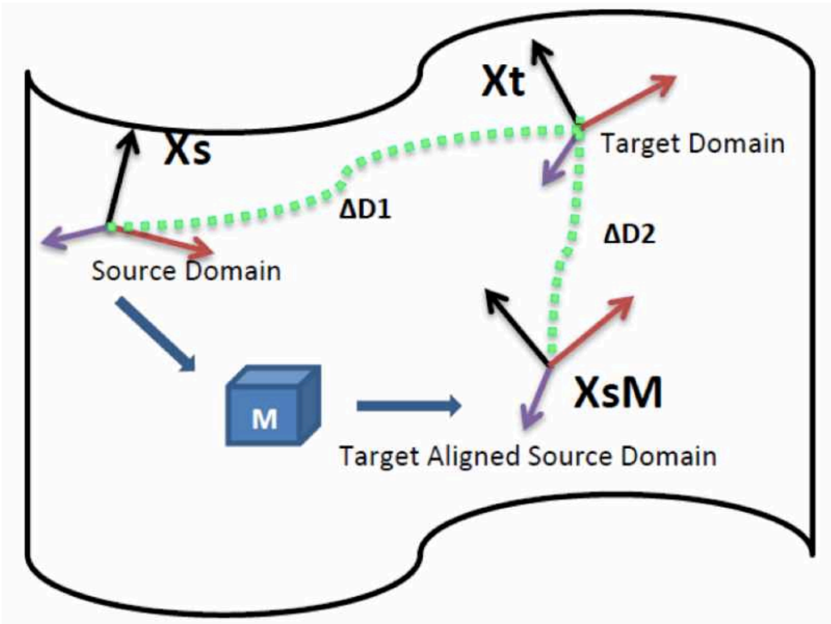
- Optimizing a (linear) mapping function that transforms the source subspace into the target one
  - Assumption: both source and target input spaces are D-dimensional

1. Transform every source and target data in the form of a D-dimensional z-normalized vector (i.e. of zero mean and unit standard deviation)
2. Using **PCA**, select for each domain *d* eigenvectors (corresponding to the largest eigenvalues)
3. These eigenvectors are used as **bases** of the source and target subspaces, respectively denoted by X_S and X_T (X_S, $X_T \in \mathrm{R}^{D \times d}$).
4. Realize the subspaces **alignment**

- Alignment of the basis vectors using a transformation matrix **M** from X_S to X_T

$$F(M) = ||X_S M - X_T||_F^2$$ Frobenius norm

$$M^\star = \underset{M}{\text{Argmin}}\{F(M)\}$$



Algorithm 1: Subspace alignment DA algorithm

Data: Source data S, Target data T, Source labels Y_S, Subspace dimension d
Result: Predicted target labels Y_T
$\mathbf{S}_1 \leftarrow PCA(S, d)$ (source subspace defined by the first d eigenvectors) ;
$\mathbf{S}_2 \leftarrow PCA(T, d)$ (target subspace defined by the first d eigenvectors);
$\mathbf{X}_a \leftarrow \mathbf{S}_1\mathbf{S}_1{}'\mathbf{S}_2$ (operator for aligning the source subspace to the target one);
$\mathbf{S}_a = S\mathbf{X}_a$ (new source data in the aligned space);
$\mathbf{T}_T = T\mathbf{S}_2$ (new target data in the aligned space);
$Y_T \leftarrow Classifier(\mathbf{S}_a, \mathbf{T}_T, Y_S)$;

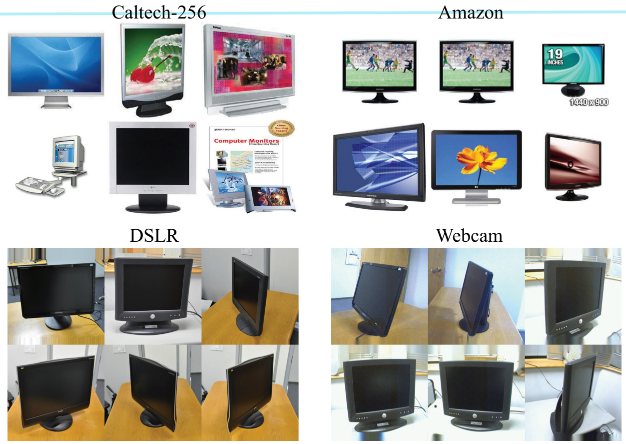
- $\mathbf{M}^* = \mathbf{S}_1{}'\mathbf{S}_2$ corresponds to the "subspace alignment matrix": $\mathbf{M}^* = \mathrm{argmin}_{\mathbf{M}} \|\mathbf{S}_1\mathbf{M} - \mathbf{S}_2\|$
- $X_a = \mathbf{S}_1\mathbf{S}_1{}'\mathbf{S}_2 = \mathbf{S}_1\mathbf{M}^*$ projects the source data to the target subspace
- A natural similarity: $\mathrm{Sim}(\mathbf{x}_s, \mathbf{x}_t) = \mathbf{x}_s\mathbf{S}_1\mathbf{M}^*\mathbf{S}_1{}'\mathbf{x}_t' = \mathbf{x}_s\mathbf{A}\mathbf{x}_t'$

Subspace alignment: empirical results

- Adaptation from Office/Caltech-10 datasets (four domains to adapt) is used as source and one as target
- Comparisons
  - Baseline 1: projection on the source subspace
  - Baseline 2: projection on the target subspace
  - 2 related methods : GFK [Gong et al., CVPR'12] and GFS [Gopalan et al.,ICCV'11]

...

Subspace alignment: empirical results



Method	C→A	D→A	W→A	A→C	D→C	W→C
Baseline 1	44.3	36.8	32.9	36.8	29.6	24.9
Baseline 2	44.5	38.6	34.2	37.3	31.6	28.4
GFK	44.8	37.9	37.1	38.3	31.4	29.1
OUR	**46.1**	**42.0**	**39.3**	**39.9**	**35.0**	**31.8**

Method	A→D	C→D	W→D	A→W	C→W	D→W
Baseline 1	36.1	38.9	73.6	**42.5**	34.6	75.4
Baseline 2	32.5	35.3	73.6	37.3	34.2	80.5
GFK	37.9	36.1	74.6	39.8	34.9	79.1
OUR	**38.8**	**39.4**	**77.9**	39.6	**38.9**	**82.3**

Recognition accuracy using a SVM classifier

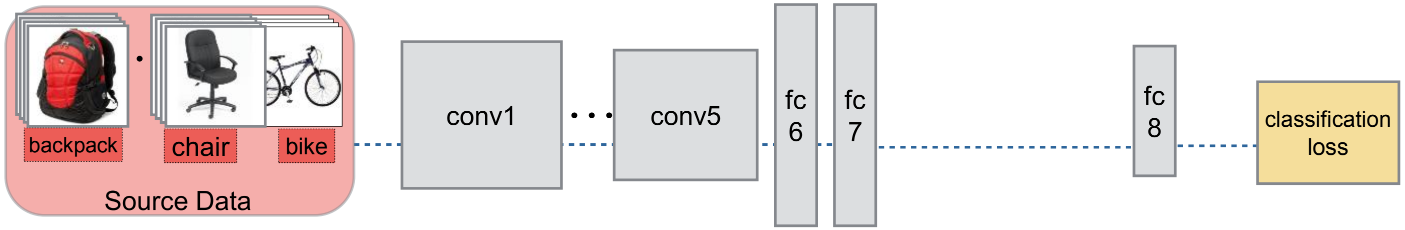
Remark1: **not** symmetrical!

Remark2: **not** that **impressive**!

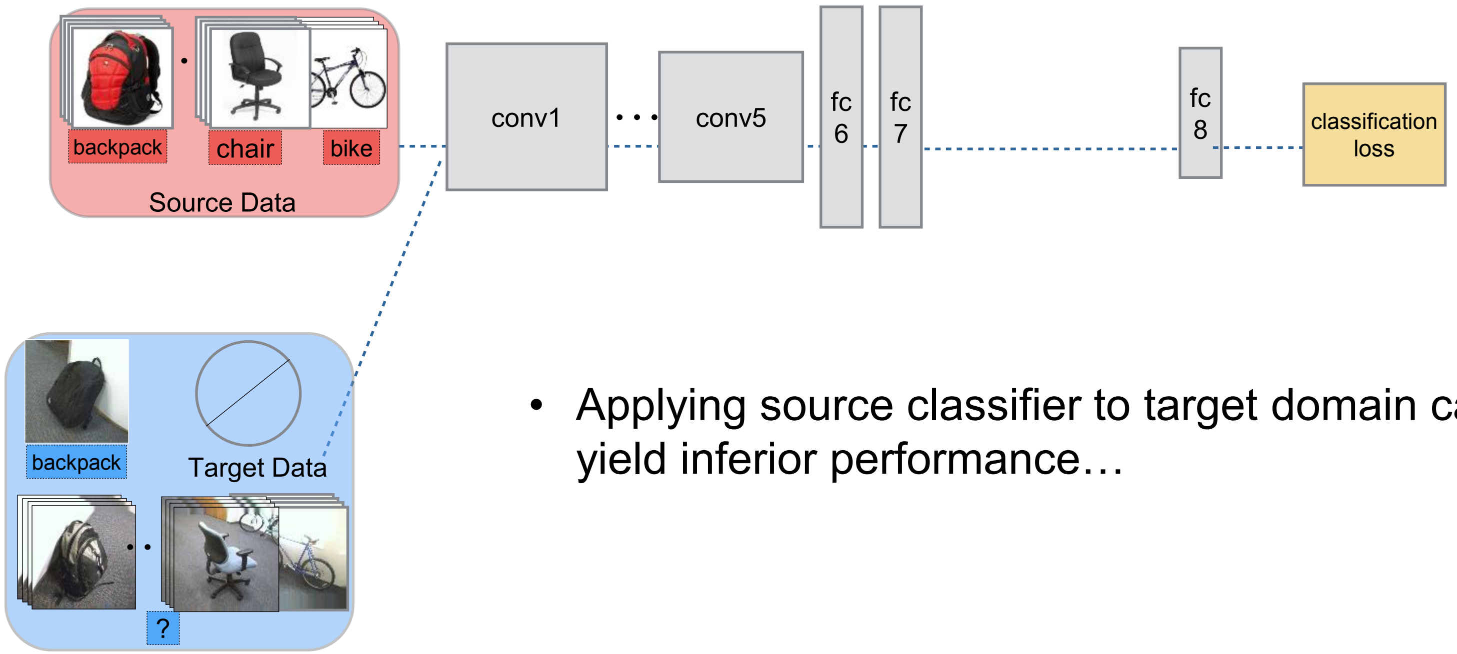
[Fernando, B., Habrard, A., Sebban, M., & Tuytelaars, T. (2013). **Unsupervised visual domain adaptation using subspace alignment**. In *Proceedings of the IEEE international conference on computer vision* (pp. 2960-2967).]

Unsupervised Domain Adaptation with **deep NNs**

Mono-task

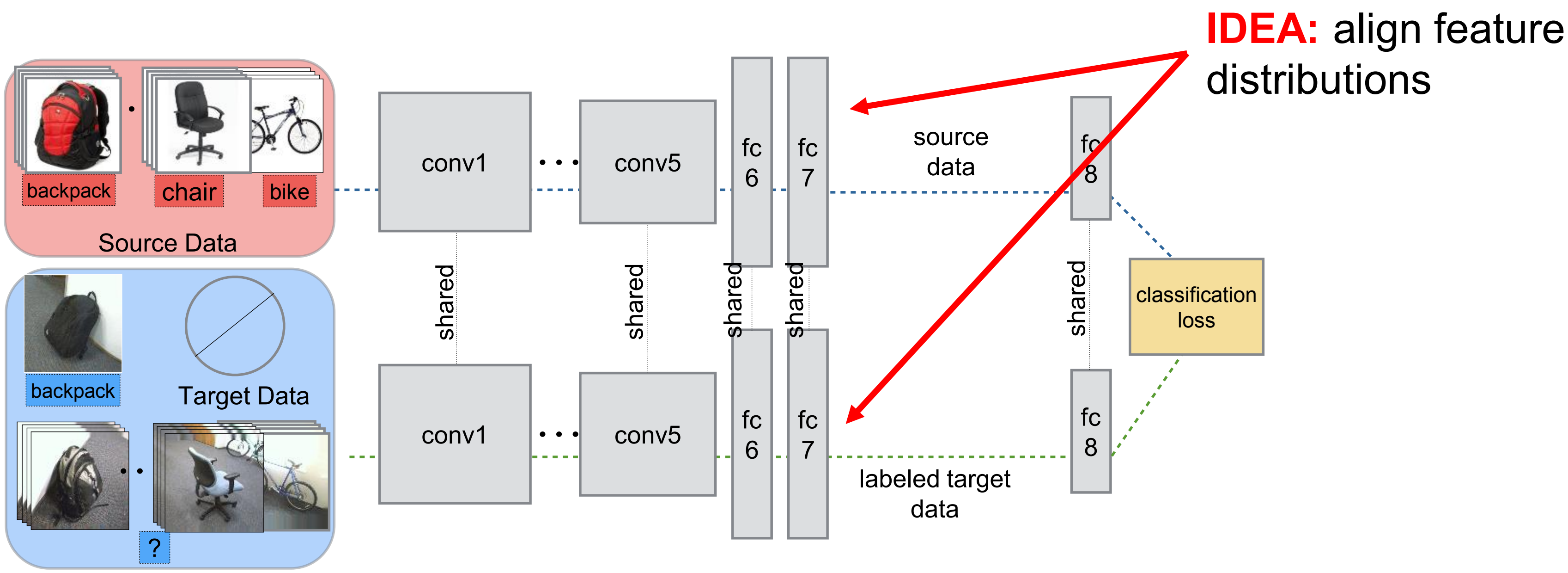


Domain adaptation



- Applying source classifier to target domain can yield inferior performance…

Unsupervised Domain Adaptation with **deep NNs**

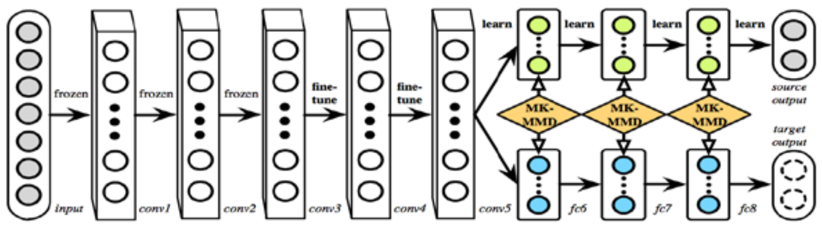


From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

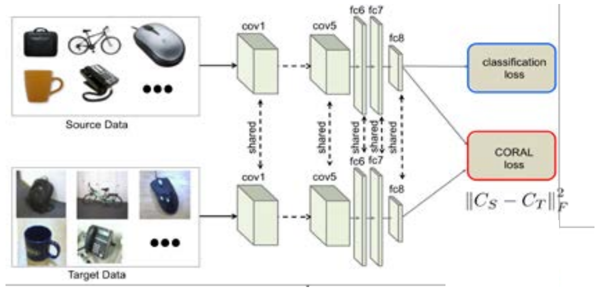
Unsupervised Domain Adaptation with **deep NNs**

Several approaches

- by minimizing distance between distributions, e.g.

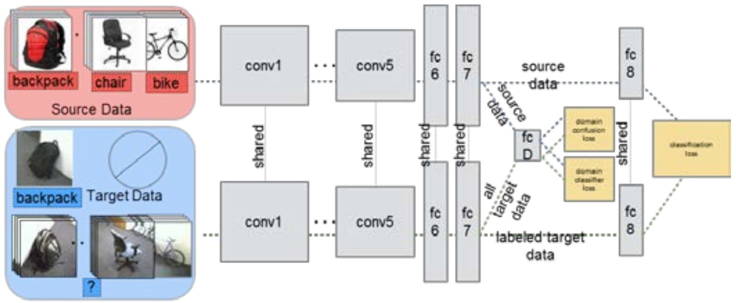


Maximum Mean Discrepancy M. Long, et al. ICML 2015

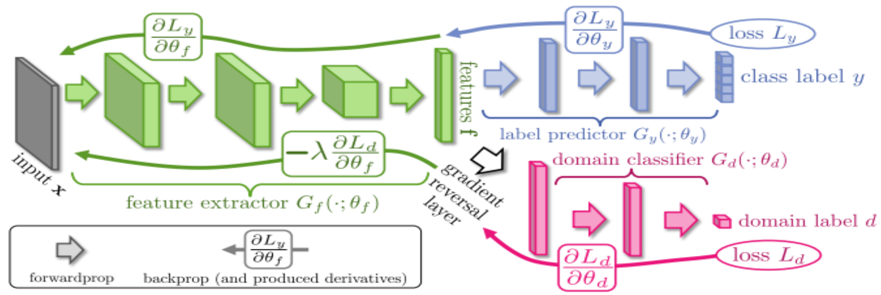


CORrelation ALignment Sun and Saenko, AAAI 2016

- …or by adversarial domain alignment, e.g.



Domain Confusion E. Tzeng et al. ICCV 2015



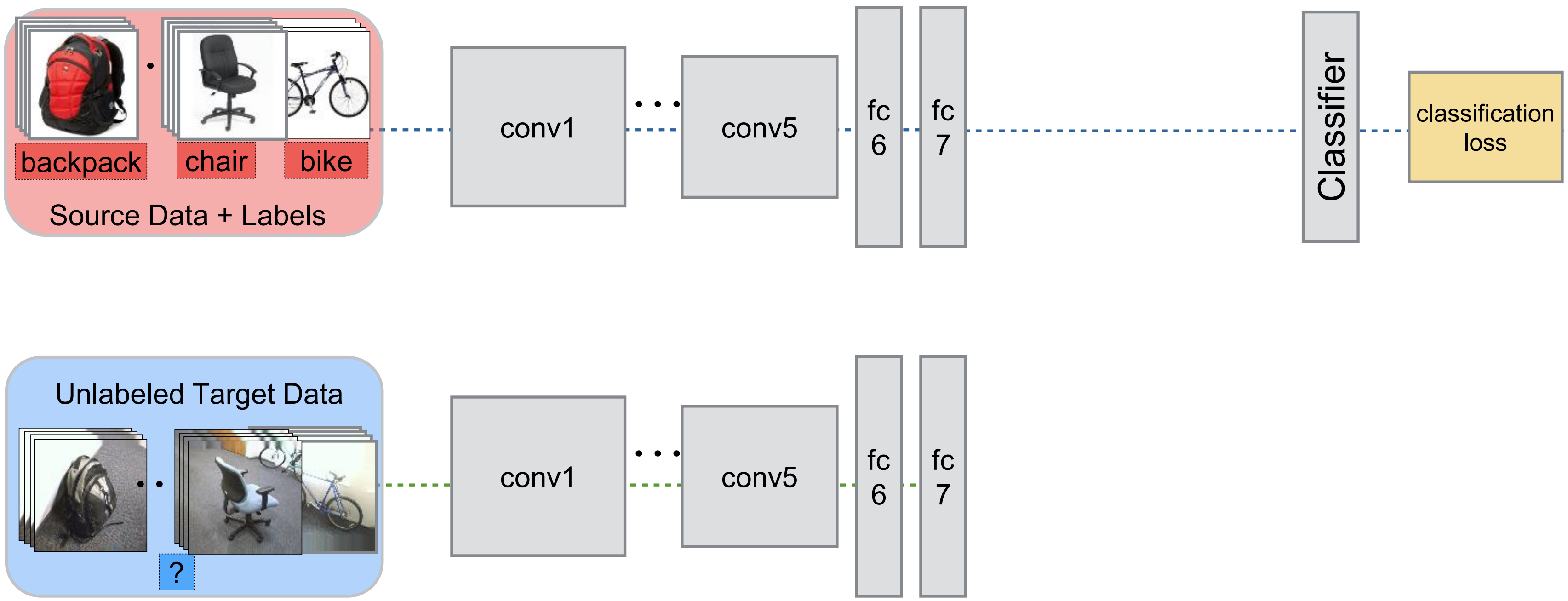
Reverse Gradient Y. Ganin and V. Lempitsky ICML 2015

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Adversarial domain adaptation

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

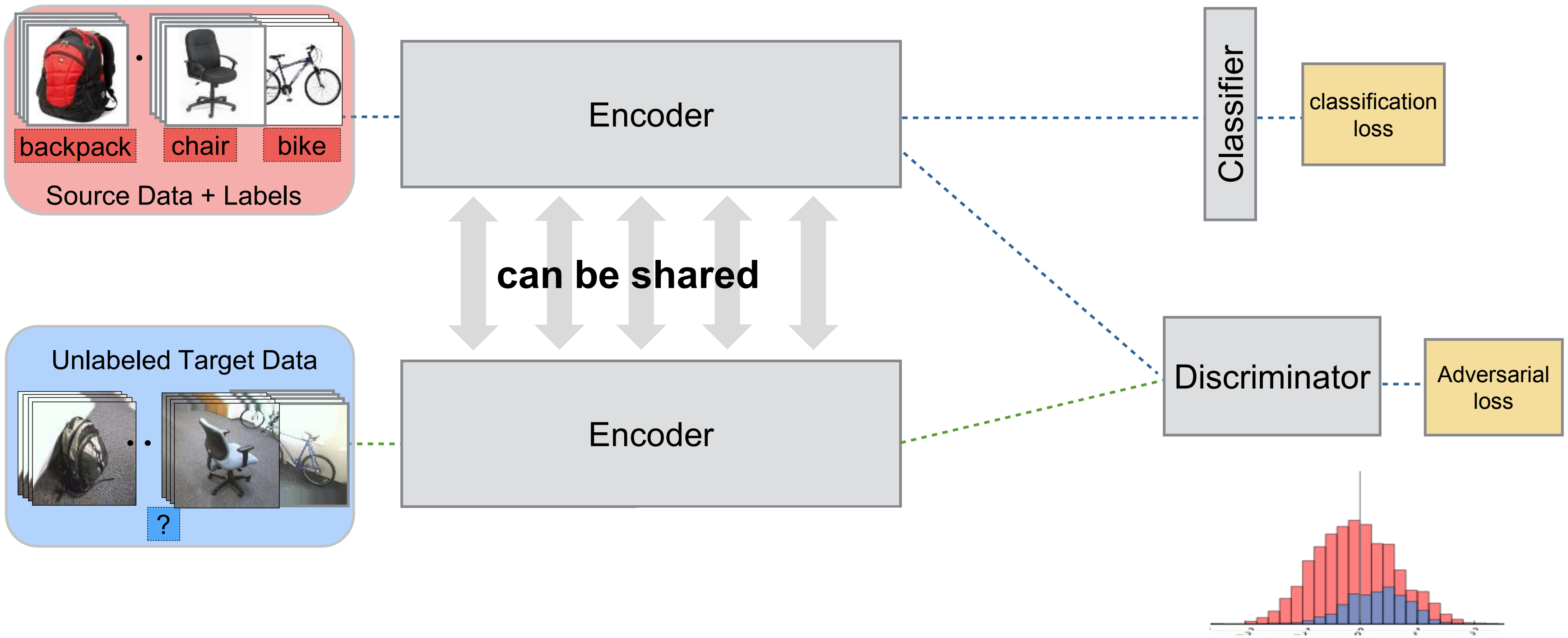
Adversarial domain adaptation



From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

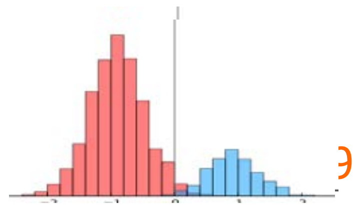
Unsupervised Domain Adaptation with **deep NNs**

Adversarial domain adaptation



One agent tries to make the distributions look alike through the encodings

The other tries to discriminate them



From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Unsupervised Domain Adaptation with **deep NNs**

Adversarial domain adaptation

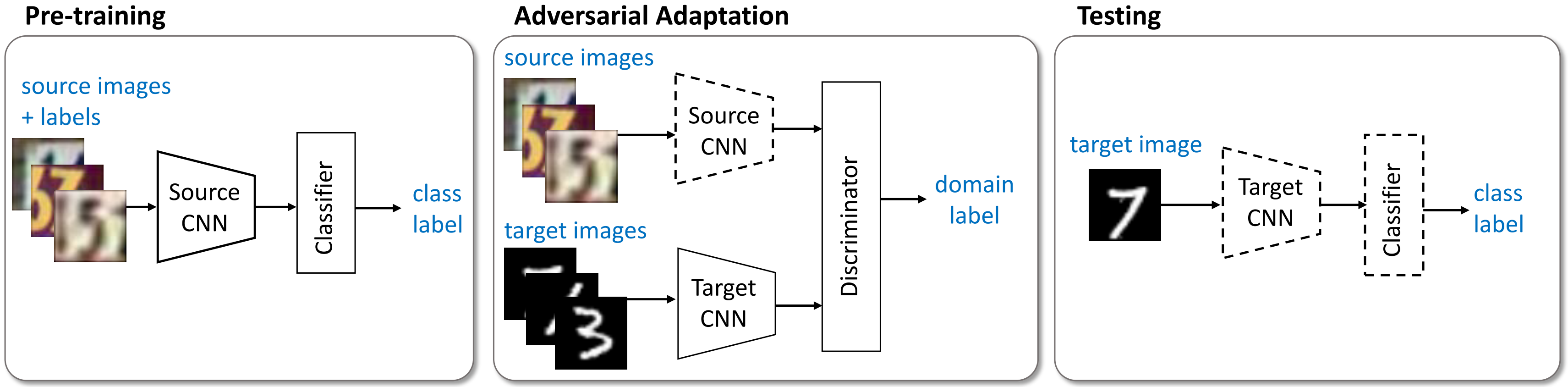


Figure 3: An overview of our proposed Adversarial Discriminative Domain Adaptation (ADDA) approach. We first pre-train a source encoder CNN using labeled source image examples. Next, we perform adversarial adaptation by learning a target encoder CNN such that a discriminator that sees encoded source and target examples cannot reliably predict their domain label. During testing, target images are mapped with the target encoder to the shared feature space and classified by the source classifier. Dashed lines indicate fixed network parameters.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7167-7176).

Unsupervised Domain Adaptation with **deep NNs**

Adversarial domain adaptation

Method	MNIST → USPS	USPS → MNIST	SVHN → MNIST
Source only	0.752 ± 0.016	0.571 ± 0.017	0.601 ± 0.011
Gradient reversal	0.771 ± 0.018	0.730 ± 0.020	0.739 [19]
Domain confusion	0.791 ± 0.005	0.665 ± 0.033	0.681 ± 0.003
CoGAN	0.912 ± 0.008	0.891 ± 0.008	did not converge
ADDA (Ours)	0.894 ± 0.002	0.901 ± 0.008	0.760 ± 0.018

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Outline

1. Classes severely unbalanced
2. Learning from positive examples only
3. Semi-supervised learning
4. Active learning
5. Domain adaptation
6. Tracking

Tracking: an intriguing idea

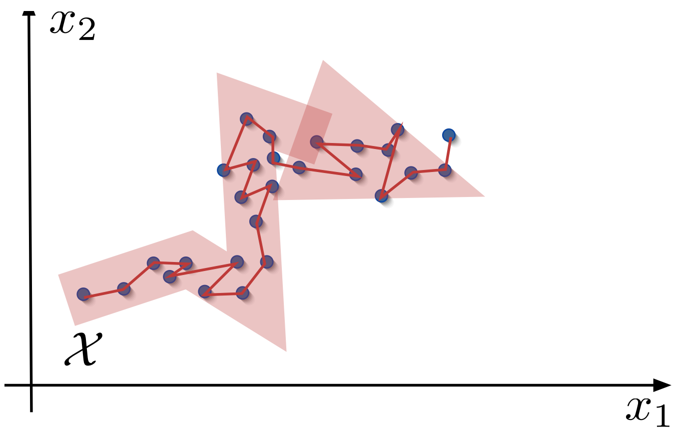
[Richard Sutton, Anna Koop & David Silver (2007). *On the role of tracking in stationary environments*. ICML-2007]

Even in stationary environments, it **can be advantageous to act as if the environment was changing**!!!

Tracking: an intriguing idea

In a lot of natural settings:

- Data comes *sequentially*
- *Temporal consistency*: consecutive data points come from “similar” distribution: not i.i.d.



This enables:

- Powerful learning
- with limited resources (time + memory)

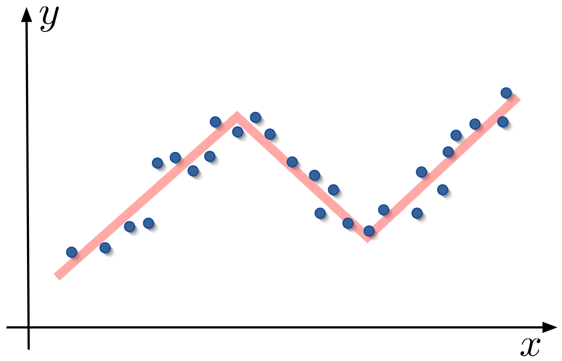
SKS:07 R. Sutton and A. Koop and D. Silver (2007) *“On the role of tracking in stationary environments”* (ICML-07) Proceedings of the 24th international conference on Machine learning, ACM, pp.871-878, 2007.

...

Tracking: an intriguing idea

Assumptions:

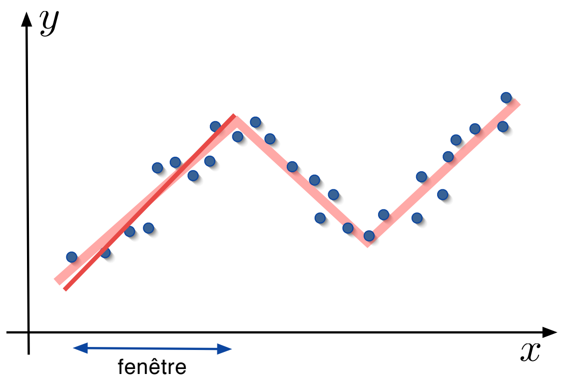
- Data streams
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.
- Limited resources: Restricted hypothesis space $\mathcal{H}$



"Local" learning

and local prediction :

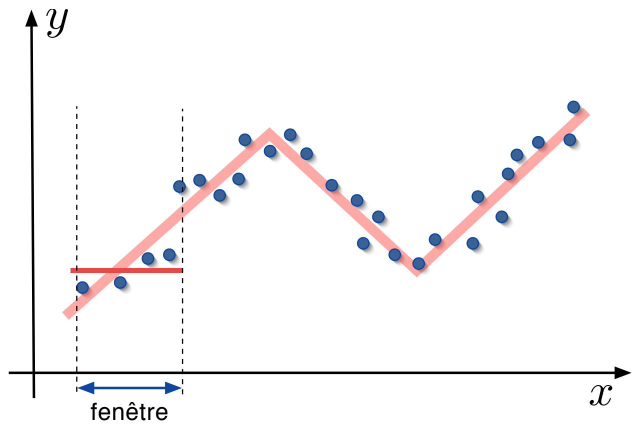
$$\begin{aligned} L_t &= \ell(h_t(\boldsymbol{x}_t), y_t) \\ &= \ell(h_t(\boldsymbol{x}_t), f(x_t, \theta_t)) \end{aligned}$$



Tracking: an intriguing idea

Assumptions:

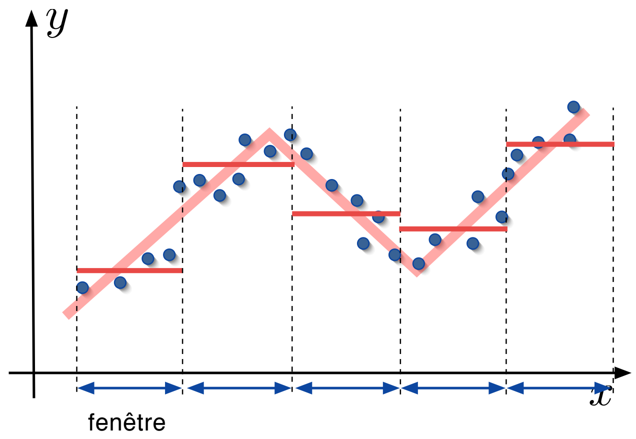
- Data streams
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.
- Limited resources: Restricted hypothesis space $\mathcal{H}$



"Local" learning

and local prediction :

$$\begin{aligned} L_t &= \ell(h_t(\boldsymbol{x}_t), y_t) \\ &= \ell(h_t(\boldsymbol{x}_t), f(x_t, \theta_t)) \end{aligned}$$



Tracking in stationary environments

- A toy environment

$$y_t = \frac{1}{1 + e^{-w_t x_t}}$$

$$w_{t+1} = w_t + \alpha\,(z_t - y_t)\,x_t$$

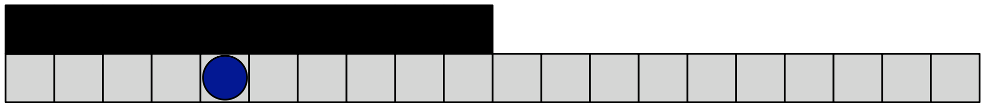


Figure 1. The Black and White world. The agent follows a random walk right and left, occasionally observing the color above it. The states wrap.

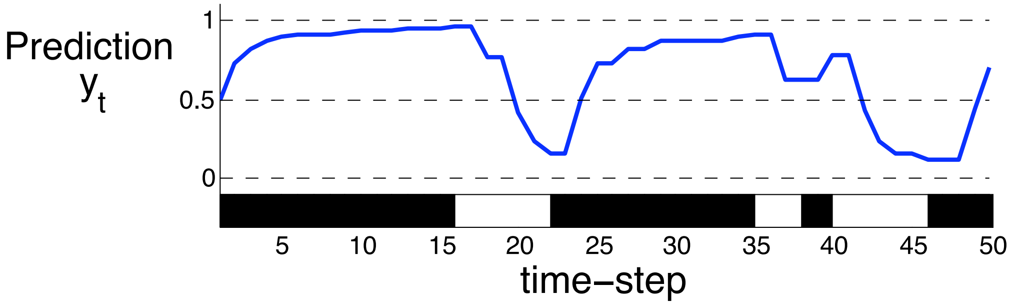
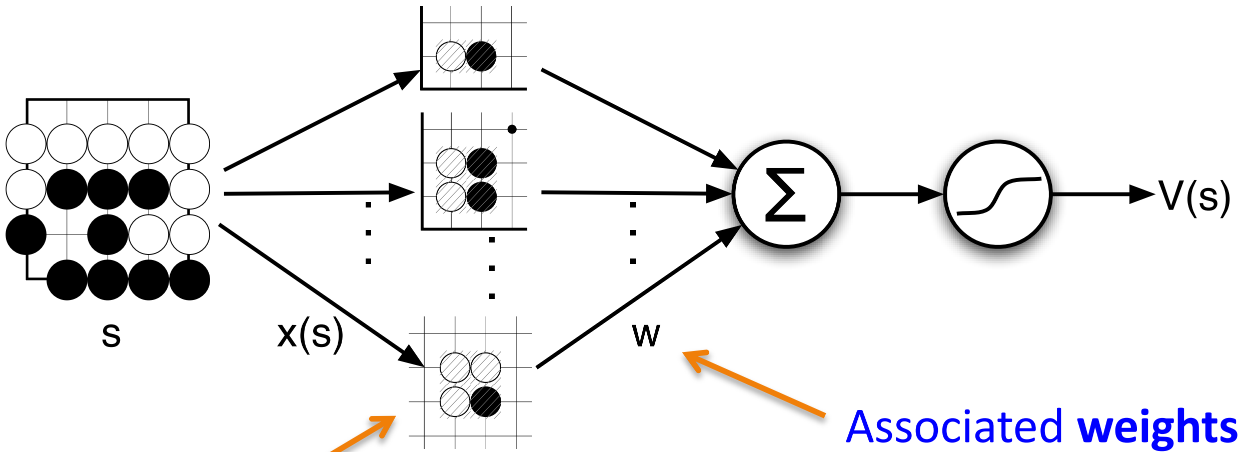


Figure 2. A sample trajectory in the Black and White world, showing the prediction on each time-step and the actual color above the agent. The prediction is modified only on time steps on which the color is observed. Here $\alpha = 2$.

Tracking to play Go

- 5 x 5 Go
  - More than 5×10^{10} unique positions
- Usual approach: learn a **general** evaluation function V(*s*) valid $\forall s$

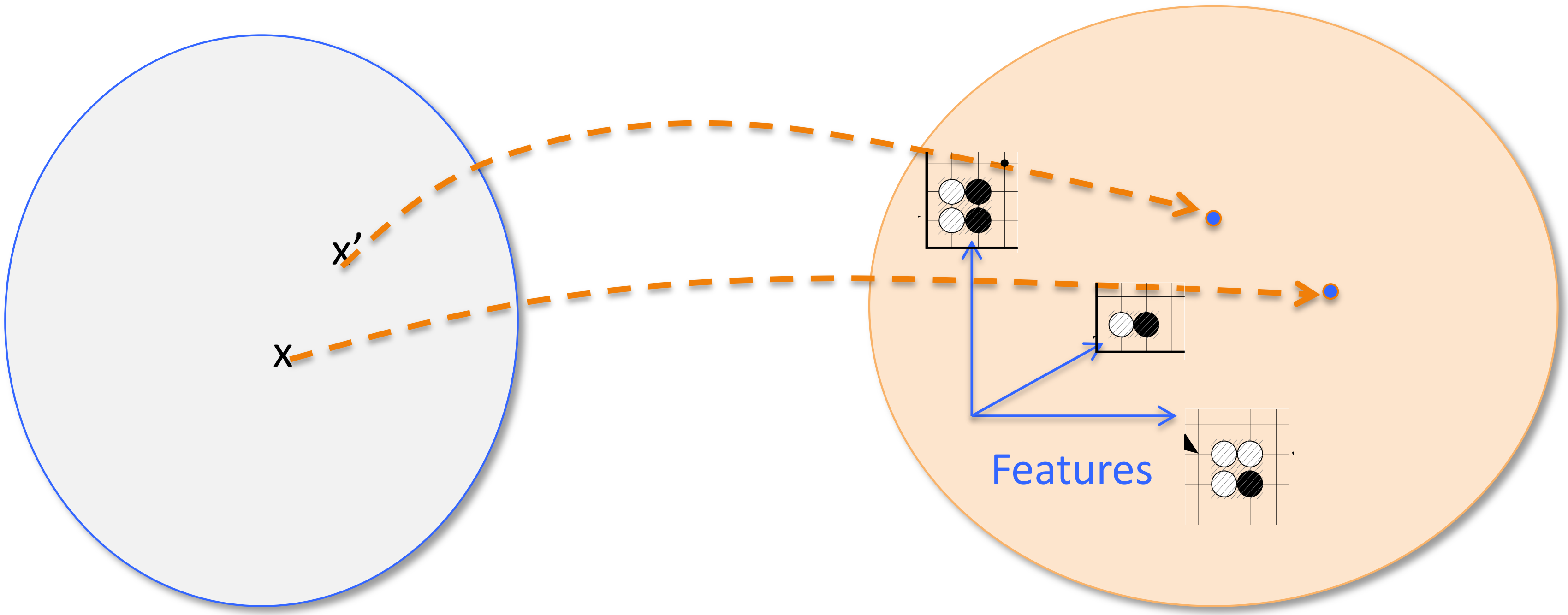


Associated **weights** (learnt)

Features describing the situation

Tracking as **local changes** of representation

…



Space of **go positions**

Embedding

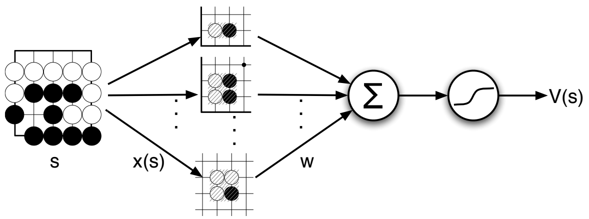
Space of representations

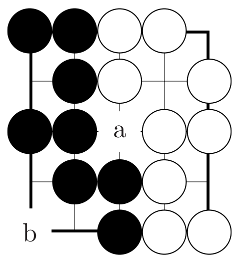
Weighted **features**

Tracking in stationary environments

- Tracking approach: learn an **evaluation** function V(*s*)

local to the current *s*

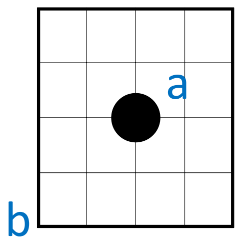




In general, playing (a) (center) is better than playing (b)

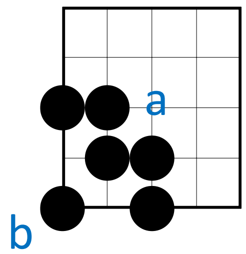
BUT

In this situation, playing (b) is better than playing (a)



More weight

More weight



Tracking in stationary environments

Tracking to play Go

Comparison:

- learn **a general evaluation function** V(s)
  - On 250,000 complete episodes of self-play
- Learn **successive evaluation functions** Vt(s) attuned to the current states
  - On 10,000 episodes of self-play starting from the current position

Features	Tracking beats converging		
	Black	White	Total
1×1	82%	43%	62.5%
2×2	90%	71%	80.5%
3×3	93%	80%	86.5%

Table 1. Percentage of 5×5 Go games won by the tracking agent playing against the converging agent when playing as Black (first to move) and as White.

Tracking in stationary environments

Comparison:

– **learn a general evaluation function** V(s)
  - On 250,000 complete episodes of self-play
– Learn **successive evaluation functions** Vt(s) attuned to the current state
  - On 10,000 episodes of self-play starting from the current position

Features	Total features	CPU (minutes) Tracking	CPU (minutes) Converging
1×1	75	3.5	10.1
2×2	1371	5.7	13.8
3×3	178518	9.1	22.2

Table 2. Memory and CPU requirements for tracking and converging agents. The total number of binary features indicates the memory consumption. The CPU time is the average training time required to play a complete game: 250,000 episodes of training for the converging agent; 10,000 episodes of training per move for the tracking agent.