Learning with a Time-Evolving Data Distribution

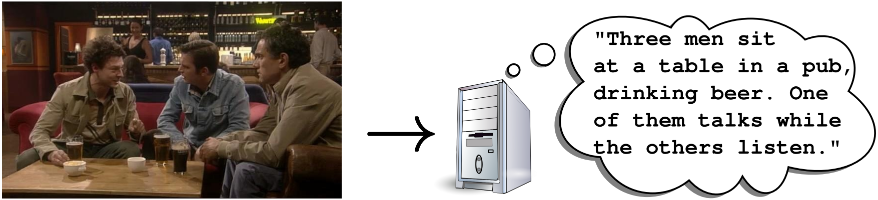
Christoph Lampert

TASK-CV Workshop at ECCV, September 12, 2014

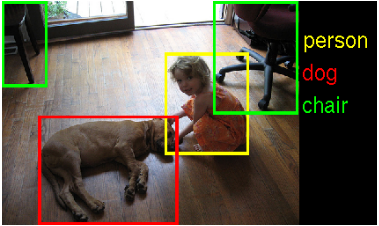
IST AUSTRIA
Institute of Science and Technology

Long Term Goal: Visual Scene Understanding

Automatic systems that can analyze and interpret visual data



Good Results under Constant Conditions...



Object Detection

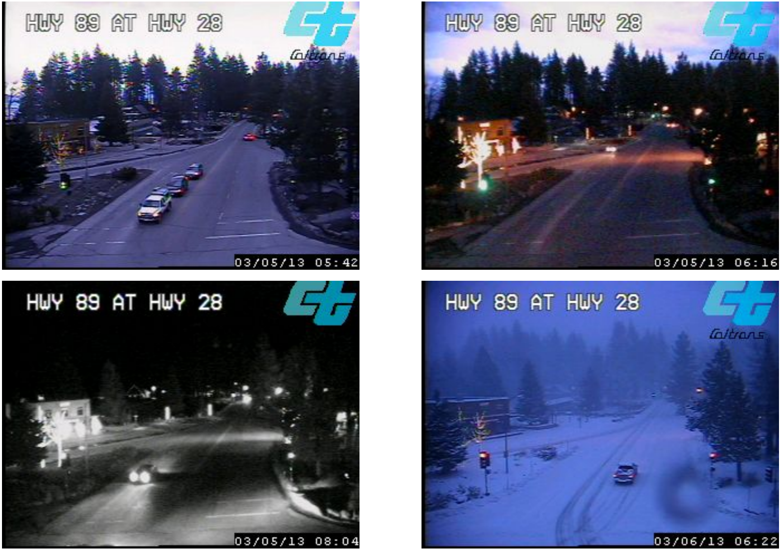
Scene Categorization

Action Classification

Object Tracking

Images: ImageNet, SUN, Hollywood, Babenko

Open Problem: Domain Shift

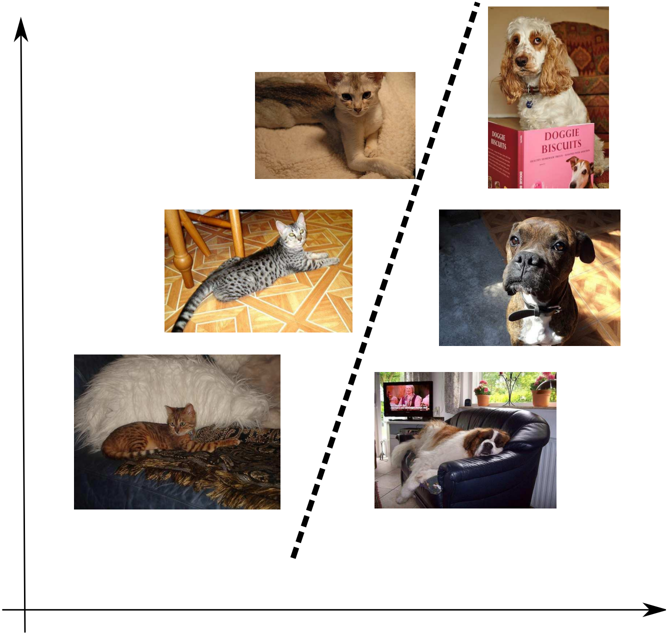


Data distribution changes between training and test time

Images: [Hofmann et. al., CVPR 2014]

A (Learning) Task

A (Learning) Task



Definition: A (Learning) Task

Task: $T = \{\mathcal{X}, \mathcal{Y}, p, S, \ell\}$

- Input set, $\mathcal{X}$, e.g. images
- Label set, $\mathcal{Y}$, e.g. "object" vs. "background"
- Data distribution: $p(x, y)$ (unknown to learner)
- Training set: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \overset{i.i.d.}{\sim} p(x, y)$
- Loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Goal: find a function $f : \mathcal{X} \to \mathcal{Y}$ with small risk,

$$\mathbb{E}_{(x,y)\sim p(x,y)}\ \ell(y, f(x))$$

Definition: A (Learning) Task

Task: $T = \{\mathcal{X}, \mathcal{Y}, p, S, \ell\}$

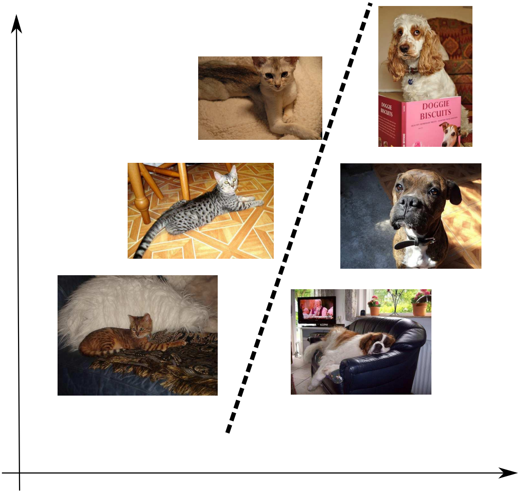
- Input set, $\mathcal{X}$, e.g. images
- Label set, $\mathcal{Y}$, e.g. `"object"` vs. `"background"`
- Data distribution: $p(x, y)$ (unknown to learner)
- Training set: $S = \{(x_1, y_1), \dots, (x_m, y_m)\} \overset{i.i.d.}{\sim} p(x, y)$
- Loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
  Think: 0/1-Loss, $\ell(y, \bar{y}) = [\![y \neq \bar{y}]\!]$ "correct" or "incorrect"

Goal: find a function $f : \mathcal{X} \to \mathcal{Y}$ with small risk,

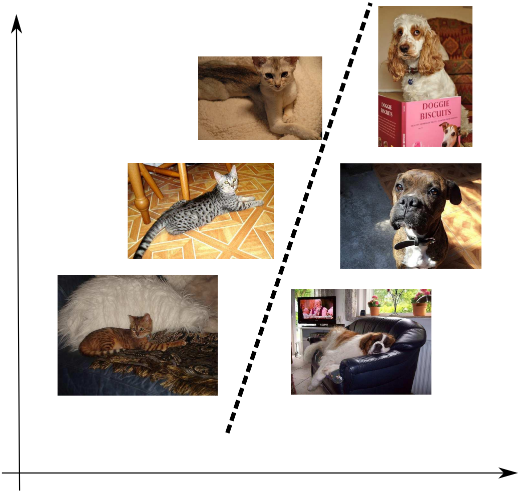
$$\mathbb{E}_{(x,y)\sim p(x,y)}\ \ell(y, f(x)) \quad = \quad \Pr_{(x,y)\sim p(x,y)} \{\ f(x) \neq y\ \}$$

Think: f makes few mistakes (at test time).

Domain Shift



Domain Shift



Definition: Domain Shift

Task: $T = \{\mathcal{X}, \mathcal{Y}, p, S\}$

- Input space, $\mathcal{X}$, e.g. images
- Output space, $\mathcal{Y}$, e.g. label: "cat" or "dog"
- Data distribution: $p(x, y)$ (unknown to learner)
- Training set: $S = \{(x_1, y_1), \dots, (x_m, y_m)\} \sim p(x, y)$

New: distribution at prediction time: $p'(x, y)$ (also unknown)

Goal: find classifier $f : \mathcal{X} \to \mathcal{Y}$ that works well at prediction time

$$\min_f \quad \mathbb{E}_{(x,y)\sim p'(x,y)} \ \ell(y, f(x))$$

Definition: Domain Shift

Task: $T = \{\mathcal{X}, \mathcal{Y}, p, S\}$

- Input space, $\mathcal{X}$, e.g. images
- Output space, $\mathcal{Y}$, e.g. label: "cat" or "dog"
- Data distribution: $p(x, y)$ (unknown to learner)
- Training set: $S = \{(x_1, y_1), \dots, (x_m, y_m)\} \sim p(x, y)$

New: distribution at prediction time: $p'(x, y)$ (also unknown)

Goal: find classifier $f : \mathcal{X} \to \mathcal{Y}$ that works well at prediction time

$$\min_f \quad \mathbb{E}_{(x,y)\sim p'(x,y)} \ \ell(y, f(x))$$

This is hopeless, unless we have additional information!

Domain Adaptation Scenarios

Supervised Domain Adaptation

- Given: (few) samples from target distribution:

$$S' = \{(x'_1, y'_1), \dots, (x'_m, y'_m)\} \ \sim \ p'(x, y)$$

Domain Adaptation Scenarios

Supervised Domain Adaptation

- Given: (few) samples from target distribution:

$$S' = \{(x'_1, y'_1), \ldots, (x'_m, y'_m)\} \;\sim\; p'(x, y)$$

Unsupervised Domain Adaptation

- Given: (many) unlabeled samples from target distribution:

$$S' = \{x'_1, \ldots, x'_m\} \;\sim\; p'(x)$$

Domain Adaptation Scenarios

Supervised Domain Adaptation

- Given: (few) samples from target distribution:

$$S' = \{(x'_1, y'_1), \dots, (x'_m, y'_m)\} \;\sim\; p'(x, y)$$

Unsupervised Domain Adaptation

- Given: (many) unlabeled samples from target distribution:

$$S' = \{x'_1, \dots, x'_m\} \;\sim\; p'(x)$$

Blind Domain Adaptation

- **no samples from target distribution**
  (but additional assumptions on the distributions)

Learning with a Time-Varying Data Distribution

Our Assumptions:

- The underlying data distribution changes smoothly over time.
- We observe samples from more than one point of time.

Examples:

- *Influenza*: every season there's slightly different viruses
- *Embedded sensors*: material fatique changes noise characteristics
- *Spam filters*: spammers adapt to countermeasures.

Learning with a Time-Varying Data Distribution

Assumptions:

- The underlying data distribution changes smoothly over time.
- We observe samples from more than one point of time.

Computer Vision Example:

- Object design evolves over time

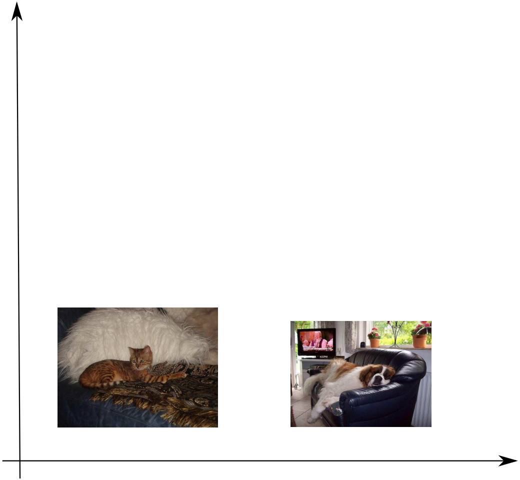
1970s

1980s

1990s

Images: [Rematas et al., ICCV VisDA, 2013]

Learning with a Time-Varying Data Distribution

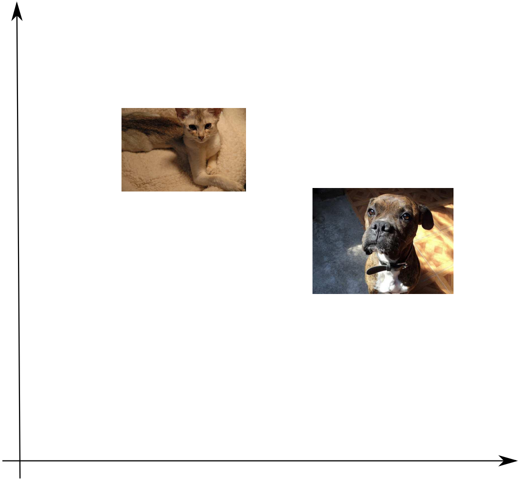


$t = 1$

Learning with a Time-Varying Data Distribution

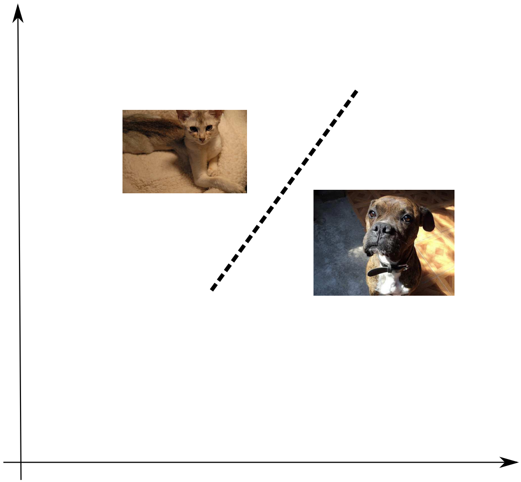
$t = 1$

Learning with a Time-Varying Data Distribution



$t = 2$

Learning with a Time-Varying Data Distribution

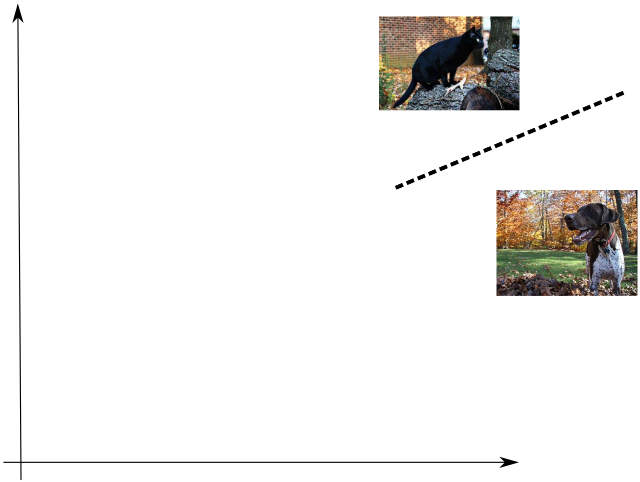


$t = 2$

Learning with a Time-Varying Data Distribution

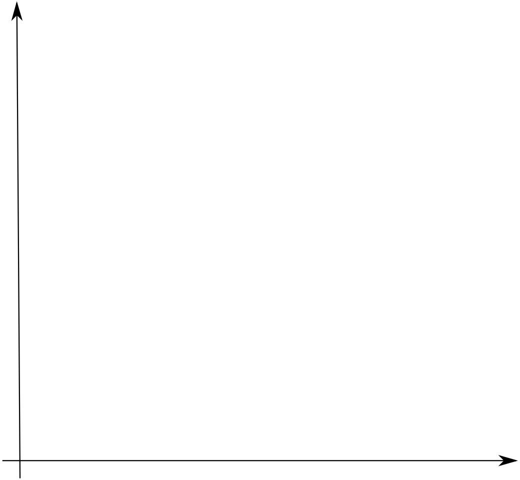
$t = 3$

Learning with a Time-Varying Data Distribution



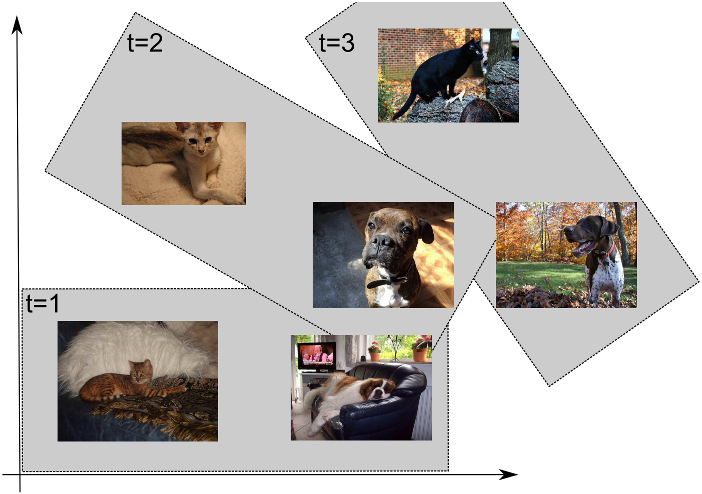
$t = 3$

Learning with a Time-Varying Data Distribution



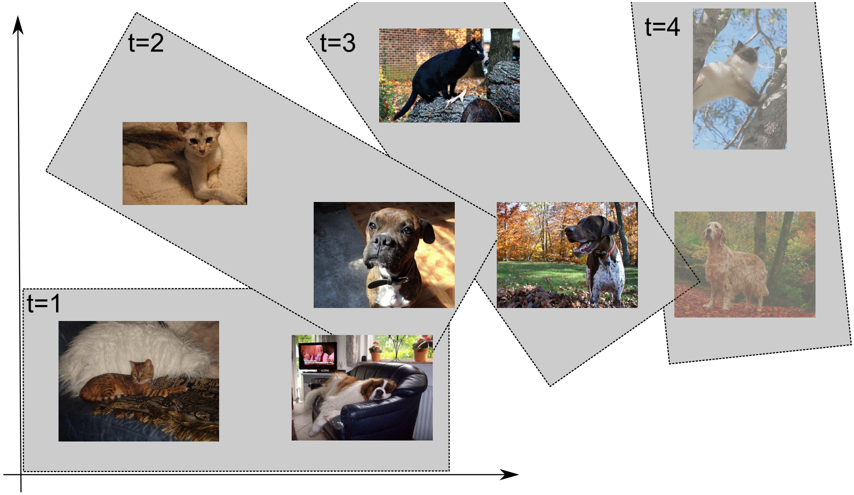
$t = 4$ (the future)

Learning with a Time-Varying Data Distribution



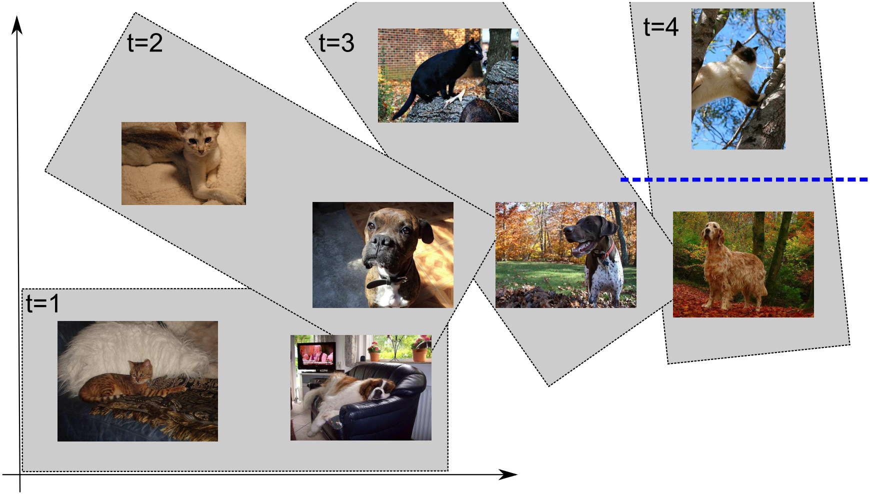
$t = 4$ (the future)

Learning with a Time-Varying Data Distribution



$t = 4$ (the future)

Learning with a Time-Varying Data Distribution



$t = 4$ (the future)

Learning with a Time-Varying Data Distribution

Task:

- Data space, $\mathcal{Z}$, e.g. images, or image/label pairs
- Time-varying data distribution: $d_t(z)$ for $t = 1, 2, \dots$
- Sample sets: $S^t = \{(z_1^t, \dots, z_{m^t}^t\} \sim d_t(z)$ for $t = 1, \dots, T$

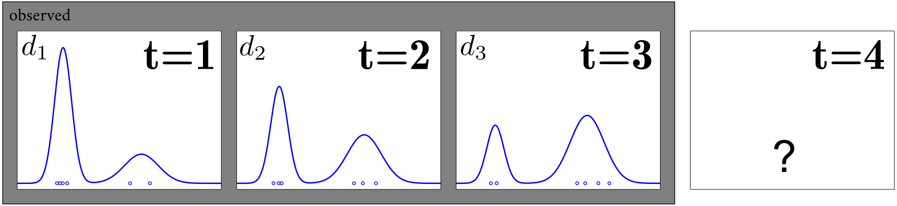
Goal: predict distribution d_{T+1} or a sample set $S^{T+1} \sim d_{T+1}$

Learning with a Time-Varying Data Distribution

Task:

- Data space, $\mathcal{Z}$, e.g. images, or image/label pairs
- Time-varying data distribution: $d_t(z)$ for $t = 1, 2, \dots$
- Sample sets: $S^t = \{(z^t_1, \dots, z^t_{m^t}\} \sim d_t(z)$ for $t = 1, \dots, T$

Goal: predict distribution d_{T+1} or a sample set $S^{T+1} \sim d_{T+1}$

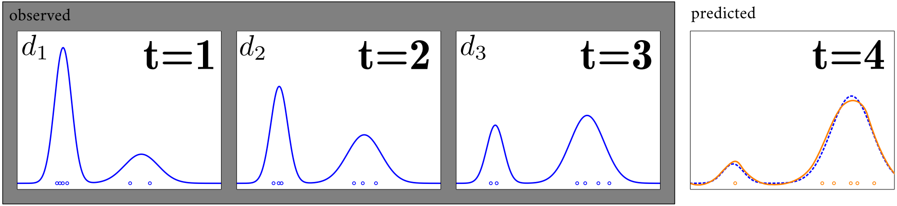


Learning with a Time-Varying Data Distribution

Task:

- Data space, $\mathcal{Z}$, e.g. images, or image/label pairs
- Time-varying data distribution: $d_t(z)$ for $t = 1, 2, \dots$
- Sample sets: $S^t = \{(z^t_1, \dots, z^t_{m^t}\} \sim d_t(z)$ for $t = 1, \dots, T$

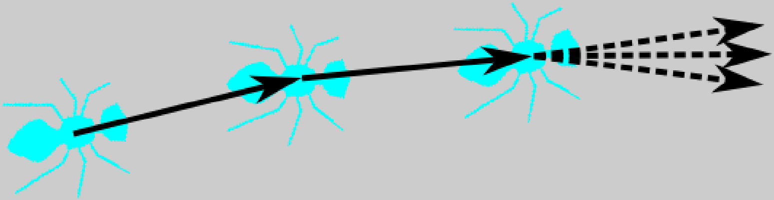
Goal: predict distribution d_{T+1} or a sample set $S^{T+1} \sim d_{T+1}$



Related Work: Motion Models for Tracking

Given: partial object trajectory

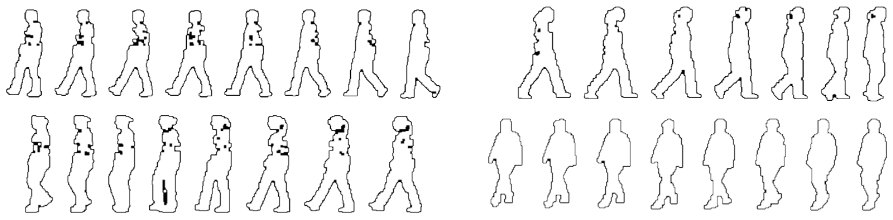
Task: predict likely next locations



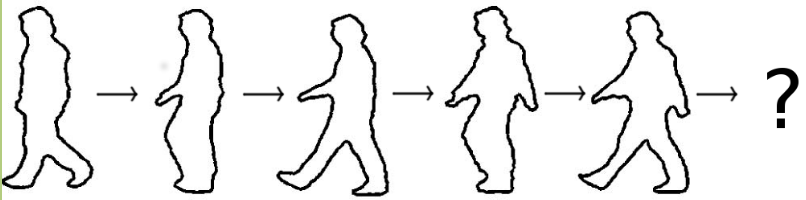
Ant image: [Khan et al, IROS 2003]

Related Work: Learning (Shape) Dynamics

Given: set of sequences



Task: learn a model that can extrapolate

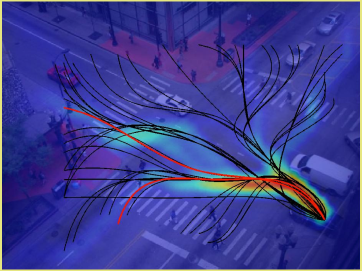


Images: [Wang et al, TPAMI 2003], [Cremers, TPAMI 2006]

Related Work: Activity Forecasting

Given: set of video sequences

Task: make long-term prediction of object movement

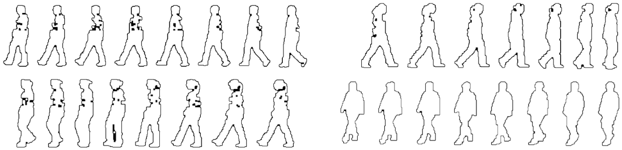


Images: [Kitani et al, ECCV 2012], [Walker et al, CVPR 2014]

What's the difference?

Learning Object Dynamics:

- training data: observations of objects changing over time



- extract variation from object corresponence between time steps

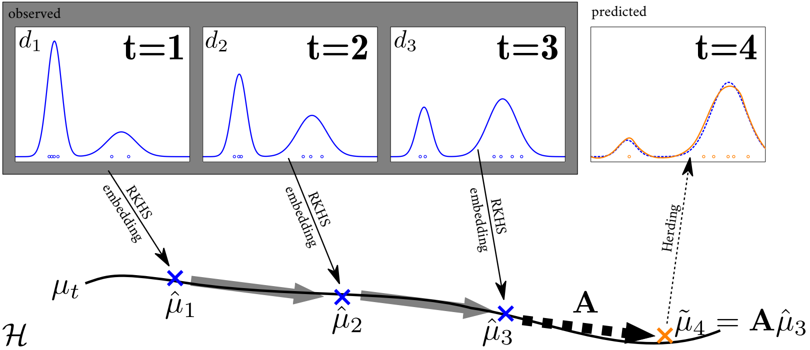
Blind Domain Adaptation:

- training data: changing distribution/populations, not individuals

- no corresponences between examples at different times

Extrapolating the Distribution Dynamics

[CHL, "Blind Domain Adaptation: An RKHS Approach", arxiv:1406.5362 [stat.ML]]



Three useful tools:

- Hilbert space embeddings of probability distributions [Smola et al., ALT 2007]
- Vector-valued regression [Micchelli & Pontil, Neural Computation 2005]
- Kernel Herding [Chen et al., UAI 2010]

Hilbert Space Embeddings of Probability Distributions

[Smola et al. "A Hilbert space embedding for distributions", ALT 2007]

Notation:

- $\mathcal{Z}$, *input space*, e.g. images, or image/label pairs
- $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, *positive definite kernel function*
- $\mathcal{H}$, the induced *reproducing kernel Hilbert space (RKHS)*
- $\boldsymbol{\varphi} : \mathcal{Z} \to \mathcal{H}$, the *induced feature map*, $\boldsymbol{\varphi}(z) = k(z, \cdot)$

For any probability distribution p on $\mathcal{Z}$:

- $\mu(p) = \mathbb{E}_{z\sim p}\{\boldsymbol{\varphi}(z)\}$ mean vector embedding of p into $\mathcal{H}$

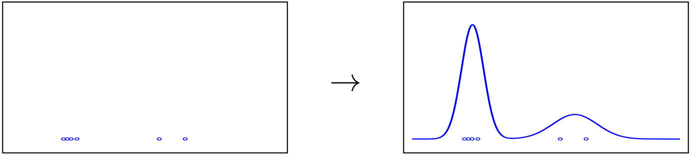
Given a set $S = \{z_1, \ldots, z_n\}$ of i.i.d. samples from p:

- $\hat{\mu}(S) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\varphi}(z_i)$ empirical mean vector embedding

Hilbert Space Embeddings of Probability Distributions

[Smola et al. "A Hilbert space embedding for distributions", ALT 2007]

Same construction as **kernel density estimation**



but result has interpretation as vector in a Hilbert space.

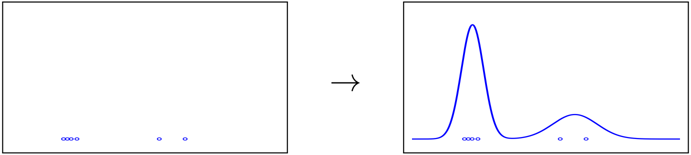
Properties:

- embedding allows us to treat distributions as vectors
- $\hat{\mu}(S_n) \to \mu(p)$ for $n \to \infty$, if $S_n = \{z_1, \dots, z_n\} \sim p$
- $\langle \hat{\mu}(S), \hat{\mu}(S') \rangle_{\mathcal{H}} = \sum_{i,j} k(z_i, z'_j)$
- $\|\hat{\mu}(S) - \hat{\mu}(S')\|^2_{\mathcal{H}}$ measures how similar S and S' are
- $\mathbb{E}_{z \sim p(z)}\{f(z)\} = \langle \mu(p), f \rangle_{\mathcal{H}}$ for $f \in \mathcal{H}$

Hilbert Space Embeddings of Probability Distributions

[Smola et al. "A Hilbert space embedding for distributions", ALT 2007]

Same construction as **kernel density estimation**



but result has interpretation as vector in a Hilbert space.

Properties:

- embedding allows us to treat distributions as vectors
- $\hat{\mu}(S_n) \to \mu(p)$ for $n \to \infty$, if $S_n = \{z_1, \dots, z_n\} \sim p$
- $\langle \hat{\mu}(S), \hat{\mu}(S') \rangle_{\mathcal{H}} = \sum_{i,j} k(z_i, z'_j)$
- $\|\hat{\mu}(S) - \hat{\mu}(S')\|^2_{\mathcal{H}}$ measures how similar S and S' are
- $\mathbb{E}_{z \sim p(z)}\{f(z)\} = \langle \mu(p), f \rangle_{\mathcal{H}}$ for $f \in \mathcal{H}$

Vector-Valued Regression

[Micchelli, Pontil, "On learning vector-valued functions", Neural Computation, 2005]

Setting:

- Given: input vectors $v_1, \ldots v_n$ with $v_i \in \mathcal{V}$
- Given: output vectors $w_1, \ldots w_n$ with $w_i \in \mathcal{W}$
- Goal: find operator $\mathbf{A} : \mathcal{V} \to \mathcal{W}$ such that $\quad Tv_i \approx w_i$

Vector-Valued Regression

[Micchelli, Pontil, "On learning vector-valued functions", Neural Computation, 2005]

Setting:

- Given: input vectors $v_1, \dots v_n$ with $v_i \in \mathcal{V}$
- Given: output vectors $w_1, \dots w_n$ with $w_i \in \mathcal{W}$
- Goal: find operator $\mathbf{A} : \mathcal{V} \to \mathcal{W}$ such that $\quad Tv_i \approx w_i$

Operator-valued least-squared regression:

- Find $\mathbf{A}$ by minimizing

$$\frac{1}{2} \sum_{i=1}^{n} \|w_i - \mathbf{A} v_i\|_{\mathcal{W}}^2 + \lambda \|\mathbf{A}\|_{\mathcal{L}(\mathcal{V},\mathcal{W})}^2$$

Vector-Valued Regression

[Micchelli, Pontil, "On learning vector-valued functions", Neural Computation, 2005]

Setting:

- Given: input vectors $v_1, \dots v_n$ with $v_i \in \mathcal{V}$
- Given: output vectors $w_1, \dots w_n$ with $w_i \in \mathcal{W}$
- Goal: find operator $\mathbf{A} : \mathcal{V} \to \mathcal{W}$ such that $\quad Tv_i \approx w_i$

Operator-valued least-squared regression:

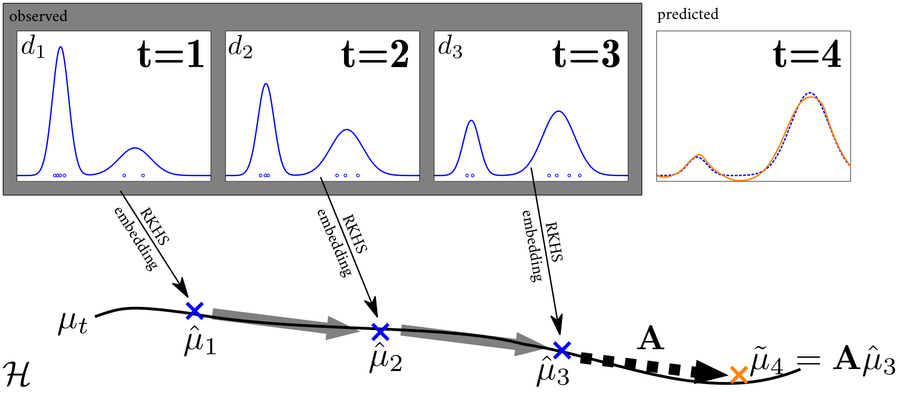
- Find $\mathbf{A}$ by minimizing

$$\frac{1}{2} \sum_{i=1}^{n} \|w_i - \mathbf{A} v_i\|_{\mathcal{W}}^2 + \lambda \|\mathbf{A}\|_{\mathcal{L}(\mathcal{V},\mathcal{W})}^2$$

Closed-form solution (similar to scalar case):

$$\mathbf{A} = \sum_{i=1}^{n} w_i \sum_{j=1}^{n} B_{ij} v_j^\top \quad \text{with} \quad B = (K + \lambda \mathrm{Id})^{-1} \text{ and } K_{ij} = \langle v_i, v_j \rangle_{\mathcal{V}}$$

Extrapolating the Distribution Dynamics



Given: sequence of embedded distributions, $\hat{\mu}_1 \to \hat{\mu}_2 \to \cdots \to \hat{\mu}_T$

Goal: predict next distribution $\hat{\mu}_{T+1}$

Extrapolating the Distribution Dynamics

Given: sample sets $S_1, \dots, S_T \subset \mathcal{Z}$, kernel $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$

Algorithm:

- form embeddings $\hat{\mu}_t = \frac{1}{n} \sum_{i=1}^{n_t} \boldsymbol{\varphi}(x_t^i)$, for $t = 1, \dots, T$
- estimate operator $\mathbf{A} : \mathcal{H} \to \mathcal{H}$ by minimizing

$$\frac{1}{2} \sum_{t=1}^{T-1} \|\hat{\mu}_{t+1} - \mathbf{A}\hat{\mu}_t\|_{\mathcal{H}}^2 + \lambda \|\mathbf{A}\|^2$$

- predict $\tilde{\mu}_{T+1}$ by applying $\mathbf{A}$ to $\hat{\mu}_T$

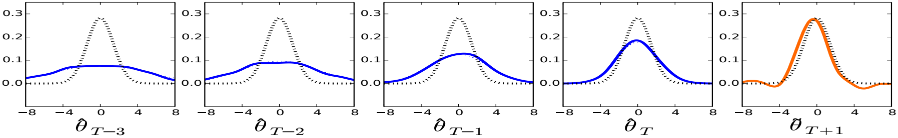
$$\tilde{\mu}_{T+1} = \mathbf{A}\hat{\mu}_T = \sum_{t=2}^{T} \beta_t \hat{\mu}_t \text{ with } \beta = (K + \lambda \mathrm{Id})^{-1} [k(S_t, S_{T+1})]_{t=1}^{T-1}$$

Observation:

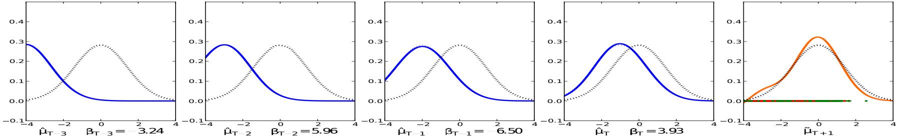
- $\tilde{\mu}_{T+1}$ consists of weighted samples from $S^1, \dots, S^T$
- weights can be positive or negative!

Extrapolating the Distribution Dynamics

Synthetic example: Gaussians with decreasing variance



Synthetic example: Gaussians with shifting mean



Training a Classifier for the Future

Predictive Domain Adaptation:

- **Given:** training sets $S_t = \{(x_1^t, y_1^t), \dots, (x_{n_t}^t, y_{n_t}^t)\}_{t=1,\dots,T}$
- **Task:** learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ for time $T+1$

Algorithm:

1) define joint kernel $k(\,(x,y),(\bar{x},\bar{y})\,) = k_{\mathcal{X}}(x,\bar{x})[\![y=\bar{y}]\!]$, where $k_{\mathcal{X}}(x,\bar{x})$ is an image kernel, e.g. χ^2.
2) predict future joint distribution $\tilde{\mu}_{T+1}$ of (x,y) in form of weights β_t^i for $t=1,\dots,T$, $i=1,\dots,n_t$.
3) learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ from weighted sample sets

Training a Classifier for the Future

Predictive Domain Adaptation:

- **Given:** training sets $S_t = \{(x_1^t, y_1^t), \dots, (x_{n_t}^t, y_{n_t}^t)\}_{t=1,\dots,T}$
- **Task:** learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ for time $T+1$

Algorithm:

1) define joint kernel $k(\,(x,y),(\bar{x},\bar{y})\,) = k_{\mathcal{X}}(x,\bar{x})[\![y=\bar{y}]\!]$, where $k_{\mathcal{X}}(x,\bar{x})$ is an image kernel, e.g. χ^2.

2) predict future joint distribution $\tilde{\mu}_{T+1}$ of (x,y) in form of weights β_t^i for $t=1,\dots,T$, $i=1,\dots,n_t$.

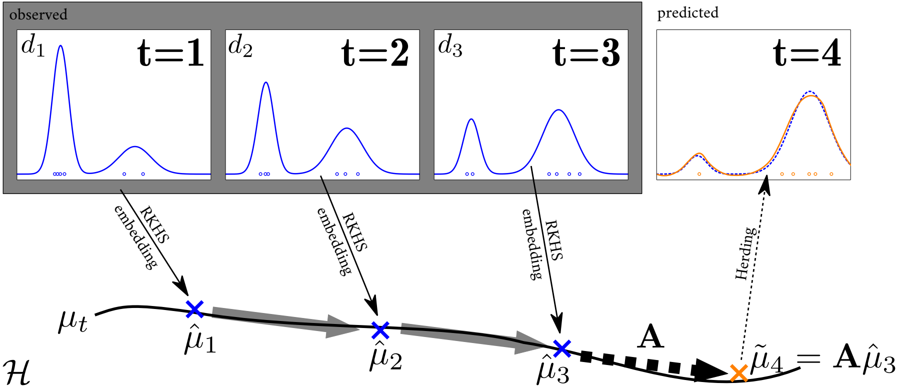
3) learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ from weighted sample sets

How?

a) some method support per-sample weights (even if negative!)

b) create a new training set according to $\tilde{\mu}_{T+1}$

Creating A Sample Set From an Embedded Distribution



(Kernel) Herding

[Chen et al., "Super-samples from kernel herding", UAI 2010], [Bach et al., "On the equivalence between herding and conditional gradient algorithms", ICML 2012]

Given: embedded distribution, $\mu \in \mathcal{H}$,

Task: find sample set, $z_1, \ldots, z_n \in \mathcal{Z}$, such that $\mu \approx \frac{1}{n} \sum_i \boldsymbol{\varphi}(z_i)$

(Kernel) Herding

[Chen et al., "Super-samples from kernel herding", UAI 2010], [Bach et al., "On the equivalence between herding and conditional gradient algorithms", ICML 2012]

Given: embedded distribution, $\mu \in \mathcal{H}$,

Task: find sample set, $z_1, \dots, z_n \in \mathcal{Z}$, such that $\mu \approx \frac{1}{n} \sum_i \boldsymbol{\varphi}(z_i)$

Idea: minimize $\| \mu - \frac{1}{n} \sum_i \boldsymbol{\varphi}(z_i) \|^2_{\mathcal{H}}$ over all $(z_1, \dots, z_n) \in \mathcal{Z}^n$.

Herding $\equiv$ Greedy Minimization

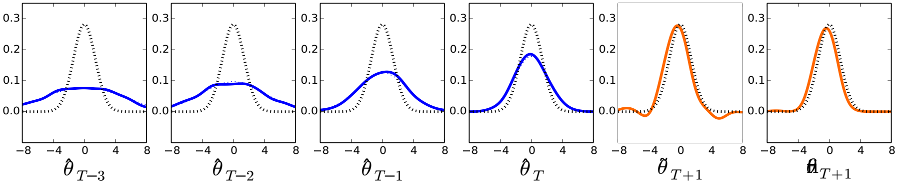
- $z_1 = \operatorname*{\mathbf{argmax}}_{z \in \mathcal{Z}} \; \left\langle \boldsymbol{\varphi}(z), \mu \right\rangle_{\mathcal{H}}$
- for $i = 2, \dots, n$:

$$z_i = \operatorname*{\mathbf{argmax}}_{z \in \mathcal{Z}} \; \left\langle \boldsymbol{\varphi}(z), \mu - \frac{1}{i} \sum_{k=1}^{i-1} \boldsymbol{\varphi}(z_j) \right\rangle_{\mathcal{H}}$$

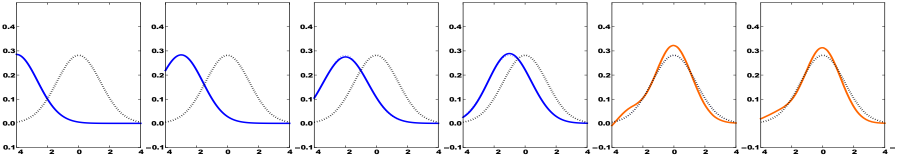
Caveat: $\mathbf{argmax}_{z \in \mathcal{Z}}$ might not easily computable.

Experiments

Synthetic example: Gaussians with decreasing variance



Synthetic example: Gaussians with shifting mean



Blind Domain Adaptation: CarEvolution dataset [1]

- 3 classes, 1086 images in 4 groups: *1970s, 1980s, 1990s, 2000s*

BMW

Mercedes

VW

Accuracy (SVM)	Fisher Vectors	DeCAF features
1970s → 2000s	39.3%	38.2%
1980s → 2000s	43.8%	48.4%
1990s → 2000s	49.0%	52.4%
all → 2000s	51.2%	52.1%
proposed (temporal order)	**51.5%**	**56.2%**

[1] [Rematas et al, "Does Evolution cause a Domain Shift?", ICCV VisDA, 2013]

Experiments

Blind Domain Adaptation: CarEvolution dataset [1]

- 3 classes, 1086 images in 4 groups: *1970s, 1980s, 1990s, 2000s*

BMW

Mercedes

VW

Accuracy (SVM)	Fisher Vectors	DeCAF features
2010s $\rightarrow$ 1970s	33.5%	34.0%
2000s $\rightarrow$ 1970s	31.6%	42.7%
1990s $\rightarrow$ 1970s	46.1%	46.6%
1980s $\rightarrow$ 1970s	44.7%	33.5%
all $\rightarrow$ 1970s	46.1%	49.0%
proposed (inverse order)	**48.5%**	**54.4%**

[1] [Rematas et al, "Does Evolution cause a Domain Shift?", ICCV VisDA, 2013]

Final words...

Summary:

- **Ordinary supervised learning:** well understood, few surprises
- **Learning with changing data distributions:** many open problems!
- **Reading the machine learning literature** can be inspiring!

Final words...

Summary:

- **Ordinary supervised learning:** well understood, few surprises
- **Learning with changing data distributions:** many open problems!
- **Reading the machine learning literature** can be inspiring!

Thanks to Funding Sources:

Open Positions at IST Austria (Postdoc, PhD, Internships):
visit `www.ist.ac.at/~chl`, or send email to `chl@ist.ac.at`