CSE 291: Domain Adaptation in Vision
Manmohan Chandraker

Lecture 3: Maximum Mean Discrepancy
Course details

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

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

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

• Domain Adaptation via Transfer Component Analysis
  • https://ieeexplore.ieee.org/document/5640675

• Learning Transferable Features with Deep Adaptation Networks
  • https://arxiv.org/abs/1502.02791
Papers for Fri, Jan 24

• Unsupervised Domain Adaptation with Residual Transfer Networks
  • https://arxiv.org/abs/1602.04433

• Deep Transfer Learning with Joint Adaptation Networks
  • https://arxiv.org/abs/1605.06636
Recap
Train on Source, Test on Target

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

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

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

• Align source and target by learning a domain invariant feature
  – Feature follows same distribution whether from source or target
  – Source classifier may then generalize well to target domain
  – Assumes such a representation exists
  – Assumes marginal label distributions similar: \( P_s(y) = P_t(y) \)
Domain Mapping
Domain Mapping

Training

Testing

(unlabeled) target data \rightarrow \text{Cond. GAN} \rightarrow (unlabeled) source data

(labeled) source data \rightarrow \text{Source Class.} \rightarrow \text{class label}

Cond. GAN \rightarrow (unlabeled) source data \rightarrow \text{Source Class.} \rightarrow \text{class label}
Many Flavors of Domain Invariance

Divergence
- MMD
- Correlation
- Contrastive
- Wasserstein

Adversarial
- Generative
- Non-generative

Reconstruction
- Encoder-Decoder
- GAN

Deep domain confusion (DDC)
Deep adaptation network (DAN)
Joint adaptation network (JAN)
Residual transfer network (RTN)

Generative adversarial (GAN)
DANN
ADDA
Feature augmentation

Reconstruction classification (DRCN)
Domain separation (DSN)
Deep Domain Adaptation

Learn a representation to minimize discrepancy
Generalization guarantee

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

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

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

• How to do model selection without enough information?
  – How to tune hyperparameters with no labeled information from the target task (and no tuning on the test set)?
Example: H-divergence

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

\[
h : X \to \{0, 1\} \quad d_H(D_S, D_T) = 2 \sup_{h \in \mathcal{H}} \left| \Pr_{x \sim D_S} [h(x) = 1] - \Pr_{x \sim D_T} [h(x) = 1] \right|
\]

(Error of the best domain classifier)

Target error bounded by source error and domain discrepancy

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

Adaptability: Some model should achieve low error on both source and target
Covariate Shift

• Source domain S, target domain T
• Labeled examples \{(x, y)\} in S, only unlabeled examples \{x\} in T
• Goal: find classifier \(h(x, \theta)\) to minimize the expected error in T
  \[
  \theta^* = \arg \min \mathbb{E}_{P(x,y)} [h(x, \theta) \neq y]
  \]
• Covariate shift assumption:
  \[P_S(X) \neq P_T(X), \text{ but } P_S(Y|X = x) = P_T(Y|X = x)\]
  – Marginal distributions are not the same
  – Distribution of emails for Alice might be different from those for Bob
  – Conditional distributions are the same
  – Given a specific email, probability of being spam same for Alice and Bob
• Not always satisfied, but often still a useful approximation
Model Misspecification in Adaptation

• Fit a linear classifier to source examples using logistic regression
  – Seems quite good for source data
  – Clearly sub-optimal for target data

• Not possible to find parameter $\theta$ such that, for all $x$,
  \[ P_S(Y|X = x, \theta) = P_T(Y|X = x, \theta) \]
  – Model misspecification: optimal source and target models are different.
Reweighting for Domain Adaptation

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

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

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

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

By definition of joint probability

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

Covariate shift assumption

\[
P_S(Y|X = x) = P_T(Y|X = x) \text{ for all } x
\]
Reweighting for Domain Adaptation

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

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

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

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

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

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

  – We can now obtain

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

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

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

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

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

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

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

• Easy adaptation:

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

Define $\Phi^s, \Phi^t: \mathbb{R}^F \rightarrow \mathbb{R}^{3F}$

$\Phi^s: x \mapsto \langle x, x, 0 \rangle$

$\Phi^t: x \mapsto \langle x, 0, x \rangle$

Train classifier in augmented space.
Frustratingly Easy Adaptation

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

• Task: word classification (noun, verb, determiner)

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

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

Augmented features \((x_1, x_2, x_1^s, x_2^s, x_1^t, x_2^t)\)

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

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

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

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

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

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

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

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

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

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

• Data points $x$ drawn from a space with kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}$
• $K$ can be rewritten as dot product of two vectors:
\[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} \]
Frustratingly Easy Adaptation: Kernel

- Data points $x$ drawn from a space with kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}$
- $K$ can be rewritten as dot product of two vectors:
  \[ K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} \]
- Kernel version of easy adaptation:
  \[ \Phi^s(x) = \langle \Phi(x), \Phi(x), 0 \rangle \]
  \[ \Phi^t(x) = \langle \Phi(x), 0, \Phi(x) \rangle \]
Frustratingly Easy Adaptation: Kernel

- Data points $x$ drawn from a space with kernel $K: \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}$
- $K$ can be rewritten as dot product of two vectors:
  \[
  K(x, x') = \langle \Phi(x), \Phi(x') \rangle \chi
  \]
- Kernel version of easy adaptation:
  \[
  \Phi^s(x) = \langle \Phi(x), \Phi(x), 0 \rangle \\
  \Phi^t(x) = \langle \Phi(x), 0, \Phi(x) \rangle
  \]
- For inputs from the same domain (both $s$ or both $t$):
  \[
  \tilde{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle \chi + \langle \Phi(x), \Phi(x') \rangle \chi = 2K(x, x')
  \]
- For inputs from different domains (one $s$, one $t$)
  \[
  \tilde{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle \chi = K(x, x')
  \]
Frustratingly Easy Adaptation: Kernel

- Data points $x$ drawn from a space with kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}$
- $K$ can be rewritten as dot product of two vectors:
  $$K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}}$$
- Kernel version of easy adaptation:
  $$\Phi^s(x) = \langle \Phi(x), \Phi(x), 0 \rangle$$
  $$\Phi^t(x) = \langle \Phi(x), 0, \Phi(x) \rangle$$
- For inputs from the same domain (both $s$ or both $t$):
  $$\tilde{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} + \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} = 2K(x, x')$$
- For inputs from different domains (one $s$, one $t$)
  $$\tilde{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{X}} = K(x, x')$$
- Target data has twice as much weight for target domain predictions!
Frustratingly Easy Adaptation: Deep

• Need some extension to handle real-valued inputs
Frustratingly Easy Adaptation: Deep

• Need some extension to handle real-valued inputs
• Let there be $K$ domains
  – A common CNN $\theta$ across all domains, with output vector $h_t \in \mathbb{R}^d$
  – One CNN $\theta^k$ for each domain, with output vector $h^k_t \in \mathbb{R}^d$
Frustratingly Easy Adaptation: Deep

• Need some extension to handle real-valued inputs
• Let there be $K$ domains
  – A common CNN $\theta$ across all domains, with output vector $h_t \in \mathbb{R}^d$
  – One CNN $\theta^k$ for each domain, with output vector $h^k_t \in \mathbb{R}^d$

• Model has parameters $W \in \mathbb{R}^{L \times d}$ and $W^k \in \mathbb{R}^{L \times d}$ at top layer
• Combines the CNN outputs as

$$z^k_t = [W \ W^k] \begin{bmatrix} h_t \\ h^k_t \end{bmatrix} = Wh_t + W^kh^k_t$$

  – Analogue of sparse feature augmentation: $x_{\text{aug}} = (x, 0, \ldots, x^k, \ldots, 0)$
Frustratingly Easy Adaptation: Deep

- Need some extension to handle real-valued inputs
- Let there be $K$ domains
  - A common CNN $\theta$ across all domains, with output vector $h_t \in \mathbb{R}^d$
  - One CNN $\theta^k$ for each domain, with output vector $h_t^k \in \mathbb{R}^d$

- Model has parameters $W \in \mathbb{R}^{L \times d}$ and $W^k \in \mathbb{R}^{L \times d}$ at top layer
- Combines the CNN outputs as
  \[
  z_t^k = [W W^k] \begin{bmatrix} h_t \\ h_t^k \end{bmatrix} = Wh_t + W^kh_t^k
  \]
  - Analogue of sparse feature augmentation: $x_{aug} = (x, 0, \ldots, x^k, \ldots, 0)$
- Training: maximize log-likelihood of correct predictions in data
  \[
  J(W, W^1 \ldots W^k, \theta, \theta^1 \ldots \theta^k) = \sum_t \log \left( \frac{\exp[z_t]_{ans(t)}}{\sum_{l=1}^L \exp[z_t^l]} \right)
  \]
  Ground truth label
Kernel Trick
Linear classifiers

- Find linear function to separate positive and negative examples

\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]
\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
Support Vector Machines (SVMs)

- Discriminative classifier based on optimal separating line (for 2d case)
- Maximize the margin between the positive and negative training examples
Support vector machines

- Want line that maximizes the margin.

\[ wx + b = 1 \]
\[ wx + b = 0 \]
\[ wx + b = -1 \]

\[ x_i \text{ positive } (y_i = 1) : x_i \cdot w + b \geq 1 \]
\[ x_i \text{ negative } (y_i = -1) : x_i \cdot w + b \leq -1 \]

For support vectors, \( x_i \cdot w + b = \pm 1 \)
Support vector machines

- Want line that maximizes the margin.

For support vectors, \( x_i \cdot w + b = \pm 1 \)

Distance between point and line:

\[
\frac{|x_i \cdot w + b|}{\|w\|}
\]

For support vectors:

\[
\frac{w^T x + b}{\|w\|} = \pm 1
\]

\[
M = \frac{1}{\|w\|} - \frac{-1}{\|w\|} = \frac{2}{\|w\|}
\]
Support vector machines

- Want line that maximizes the margin.

\[ w^T x + b = 1 \]
\[ w^T x + b = 0 \]
\[ w^T x + b = -1 \]

For support vectors, \( x_i \cdot w + b = \pm 1 \)

Distance between point and line:
\[
\frac{|x_i \cdot w + b|}{||w||}
\]

Therefore, the margin is \[
\frac{2}{||w||}.
\]
Finding the maximum margin line

1. Maximize margin \[ \frac{2}{\|\mathbf{w}\|} \]

2. Correctly classify all training data points:
   \[ y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 \]
   \[ y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ positive} \\ -1 & \text{if } \mathbf{x}_i \text{ negative} \end{cases} \]

Quadratic optimization problem (over all training data):

Minimize \[ \frac{1}{2} \mathbf{w}^T \mathbf{w} \]
Subject to \[ y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \]
Finding the maximum margin line

Constrained optimization problem:

Minimize \[ \frac{1}{2} w^T w \]

Subject to \( y_i(w \cdot x_i + b) \geq 1 \), for all training points \( i \)

Lagrangian formulation for unconstrained optimization:

\[
\min_{w, b} \frac{1}{2} w^T w - \sum_i \alpha_i [y_i(w \cdot x_i + b) - 1]
\]

Condition for optimum:

\[
\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0
\]

\[
\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0
\]
Learning the weights

- **Primal problem**

\[
\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \sum_i \alpha_i \left[ y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right]
\]

- **Consider the dual problem**

\[
\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad \sum_i \alpha_i y_i = 0
\]

\[
\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_i \alpha_i
\]

- **Take derivatives with respect to** \(\alpha_i\) **and set to 0**

Note: only dot products of data points appear here
Finding the maximum margin line

- Solution: \( \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \)
- Most of the \( \alpha_i \) will turn out to be 0.
- The non-zero \( \alpha_i \) correspond to support vectors.
Finding the maximum margin line

- Solution: \( w = \sum_i \alpha_i y_i x_i \)

\[
b = y_i - w \cdot x_i \quad \text{(for any support vector)}
\]

\[
w \cdot x + b = \sum_i \alpha_i y_i x_i \cdot x + b
\]

- Classification function:

\[
f(x) = \text{sign}(w \cdot x + b) = \text{sign}(\sum_i \alpha_i y_i x_i \cdot x + b)
\]

If \( f(x) < 0 \), classify as negative.
If \( f(x) > 0 \), classify as positive.

Note: only dot products of data points appear here
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:
General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \varphi(x) \]
Nonlinear SVMs

• *The kernel trick*: instead of explicitly computing the lifting transformation \( \phi(x) \), define a kernel function \( K \) such that

\[
K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)
\]

• This gives a nonlinear decision boundary in the original feature space:

\[
\sum_i \alpha_i y_i K(x_i, x) + b
\]

• Simply replace all dot products in original space with the kernel function, also expressible as a dot product:

\[
f(x) = \text{sign}(w \cdot x + b) = \text{sign}(\sum_i \alpha_i y_i x_i \cdot x + b)
\]
“Kernel trick”: Example

2-dimensional vectors \( x = [x_1 \ x_2] \).

Let \( K(x_i, x_j) = (1 + x_i^T x_j)^2 \)

Need to show that \( K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) \):

\[
K(x_i, x_j) = (1 + x_i^T x_j)^2, \\
= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} \\
= [1 \ x_{i1}^2 \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \sqrt{2} x_{i1} \sqrt{2} x_{i2}]^T \\
\quad [1 \ x_{j1}^2 \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \sqrt{2} x_{j1} \sqrt{2} x_{j2}] \\
= \varphi(x_i)^T \varphi(x_j),
\]

where \( \varphi(x) = [1 \ x_1^2 \sqrt{2} x_1 x_2 \ x_2^2 \sqrt{2} x_1 \sqrt{2} x_2] \)
Examples of kernel functions

- **Linear:**
  \[ K(x_i, x_j) = x_i^T x_j \]

- **Gaussian RBF:**
  \[ K(x_i, x_j) = -\exp\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \]

- **Histogram intersection:**
  \[ K(x_i, x_j) = \sum_k \min\{x_i(k), x_j(k)\} \]
Maximum Mean Discrepancy
Determine whether GAN is good enough
Determine whether GAN is good enough
Maximum Mean Discrepancy: Goal

- **Have:** Two collections of samples $X, Y$ from unknown distributions $P$ and $Q$.
- **Goal:** Learn distinguishing features that indicate how $P$ and $Q$ differ.
Maximum Mean Discrepancy: Divergences

\[ P \quad Q \]

\[ \frac{P}{Q} \]
Maximum Mean Discrepancy: Divergences

\[ D_\mathcal{H}(P, Q) = \sup_{g \in \mathcal{H}} |\mathbb{E}_{X \sim P} g(X) - \mathbb{E}_{Y \sim Q} g(Y)| \]

\[ D_f(P, Q) = \int_X q(x) f \left( \frac{p(x)}{q(x)} \right) dx \]

Sriperumbudur, Fukumizu, G, Schoelkopf, Lanckriet (2012)
Maximum Mean Discrepancy

• Data integration or distribution testing
  – Determine if samples X and Y are from the same distribution
  – Determine whether samples from same domain, or two different domains

• Intuition
  – Two distributions are different if some function has different expectations
  – Evaluate this function on empirical samples from the two distributions
Kernel Mean Embedding

- Let $X \subset \mathbb{R}^d$ be the data domain (set of images with $d$ pixels)
- Let $K: X \times X \to \mathbb{R}$ be symmetric positive definite kernel
  - There exists feature map $\phi: X \to \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$
Kernel Mean Embedding

- Let $X \subset \mathbb{R}^d$ be the data domain (set of images with $d$ pixels)
- Let $K: X \times X \rightarrow \mathbb{R}$ be a symmetric positive definite kernel
  - There exists feature map $\phi: \mathcal{X} \rightarrow \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$
- Let $P$ be a probability distribution defined on $X$
- Kernel mean embedding of $P$ is defined as
  \[
  \mu_P := \mathbb{E}_{x \sim P}[\phi(x)]
  \]
  - Represents the distribution $P$ as a single point in $\mathcal{H}$
Kernel Mean Embedding

- Let $X \subset \mathbb{R}^d$ be the data domain (set of images with $d$ pixels)
- Let $K : X \times X \to \mathbb{R}$ be symmetric positive definite kernel
  - There exists feature map $\phi : X \to \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$
- Let $P$ be a probability distribution defined on $X$
- Kernel mean embedding of $P$ is defined as
  \[
  \mu_P := \mathbb{E}_{x \sim P}[\phi(x)]
  \]
  - Represents the distribution $P$ as a single point in $\mathcal{H}$
  - Example: consider $X = \mathbb{R}$, $\phi(x) := (x, x^2)^\top$, $\mathcal{H} = \mathbb{R}^2$, then
    \[K(x, y) = \phi^\top(x)\phi(y) = xy + x^2y^2\]
    - Given distribution $P$, its kernel mean embedding is
      \[\mu_P = \mathbb{E}_{x \sim P}(x, x^2)^\top = \left(\mathbb{E}_{x \sim P}[x], \mathbb{E}_{x \sim P}[x^2]\right)^\top \in \mathcal{H}\]
      - A two-dimensional vector composed of the first two moments of $P$
Kernel Mean Embedding

• Determine whether two distributions $P, Q$ are the same
  – We wish to determine when $P \neq Q$ is the same as $\mu_P \neq \mu_Q$
  – If so, we can use the kernel mean embedding to compare distributions

• It is the case for the class of “characteristic kernels”
  – Example of characteristic kernel: Gaussian, $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
  – Previous one, $K(x, y) = xy + x^2y^2$, is not characteristic
  – $P$ and $Q$ can differ in third or higher moments, but not distinguishable by $\mu$

• Maximum mean discrepancy
  – Distance between kernel mean embeddings of two distributions
    $$\text{MMD}(P, Q) = \|\mu_P - \mu_Q\|_H$$
  – For characteristic kernels, $\text{MMD} = 0$ if and only if $P = Q$
Maximum Mean Discrepancy

• Data integration or distribution testing
  – Determine if samples X and Y are from the same distribution
  – Determine whether samples from same domain, or two different domains

• Intuition
  – Two distributions are different if some function has different expectations
  – Evaluate this function on empirical samples from the two distributions
Maximum Mean Discrepancy

• Data integration or distribution testing
  – Determine if samples X and Y are from the same distribution
  – Determine whether samples from same domain, or two different domains

• Intuition
  – Two distributions are different if some function has different expectations
  – Evaluate this function on empirical samples from the two distributions

• Maximum mean discrepancy (and its empirical estimate)
  – Let F be class of functions, probability distributions p, q, with samples $x_i, y_j$

    \[
    \text{MMD} [\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} (\mathbb{E}_p[f(x)] - \mathbb{E}_q[f(y)])
    \]

    \[
    \text{MMD} [\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left( \frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{i=1}^{n} f(y_i) \right)
    \]
  – If F rich enough, MMD vanishes if and only if $p = q$
  – If F is too rich, empirical estimate incorrect due to “overfitting” to samples
Maximum Mean Discrepancy

- Let $H$ be a reproducing kernel Hilbert space
  - There exists a feature map $\phi: X \to H$
  - MMD is then the distance between mean embeddings of features
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\phi(X)] - \mathbb{E}_{Y \sim Q}[\phi(Y)] \|_H
    \]
Maximum Mean Discrepancy

• Let $H$ be a reproducing kernel Hilbert space
  – There exists a feature map $\varphi : X \rightarrow H$
  – MMD is then the distance between mean embeddings of features
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\varphi(X)] - \mathbb{E}_{Y \sim Q}[\varphi(Y)] \|_H
    \]

• Examples:
  – Let $\mathcal{X} = \mathcal{H} = \mathbb{R}^d$ and $\varphi(x) = x$, then MMD is distance between means
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\varphi(X)] - \mathbb{E}_{Y \sim Q}[\varphi(Y)] \|_H
    \]
    \[
    = \| \mathbb{E}_{X \sim P}[X] - \mathbb{E}_{Y \sim Q}[Y] \|_{\mathbb{R}^d}
    \]
    \[
    = \| \mu_P - \mu_Q \|_{\mathbb{R}^d},
    \]
Maximum Mean Discrepancy

• Let $H$ be a reproducing kernel Hilbert space
  – There exists a feature map $\varphi: X \to H$
  – MMD is then the distance between mean embeddings of features
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\varphi(X)] - \mathbb{E}_{Y \sim Q}[\varphi(Y)] \|_H
    \]

• Examples:
  – Let $\mathcal{X} = H = \mathbb{R}^d$ and $\varphi(x) = x$, then MMD is distance between means
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\varphi(X)] - \mathbb{E}_{Y \sim Q}[\varphi(Y)] \|_H
    = \| \mathbb{E}_{X \sim P}[X] - \mathbb{E}_{Y \sim Q}[Y] \|_{\mathbb{R}^d}
    = \| \mu_P - \mu_Q \|_{\mathbb{R}^d},
    \]
  – Let $\mathcal{X} = \mathbb{R}^d$ and $H = \mathbb{R}^p$, with $\varphi(x) = A'x$, where $A$ is a $d \times p$ matrix, then MMD is distance between projections of means
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P}[\varphi(X)] - \mathbb{E}_{Y \sim Q}[\varphi(Y)] \|_H
    = \| \mathbb{E}_{X \sim P}[A'X] - \mathbb{E}_{Y \sim Q}[A'Y] \|_{\mathbb{R}^p}
    = \| A' \mathbb{E}_{X \sim P}[X] - A' \mathbb{E}_{Y \sim Q}[Y] \|_{\mathbb{R}^p}
    = \| A'(\mu_P - \mu_Q) \|_{\mathbb{R}^p}.
    \]
Maximum Mean Discrepancy

• For other features, MMD can construct stronger distances
  – Let $\mathcal{X} = \mathbb{R}$ and $\varphi(x) = (x, x^2)$, then MMD can distinguish distributions with different means and different variances

\[
\text{MMD}(P, Q) = \|E_X \sim P[\varphi(X)] - E_Y \sim Q[\varphi(Y)]\|_H = \sqrt{(E_X - E_Y)^2 + (E_X^2 - E_Y^2)^2}
\]
Maximum Mean Discrepancy

• For other features, MMD can construct stronger distances
  – Let \( \mathcal{X} = \mathbb{R} \) and \( \varphi(x) = (x, x^2) \), then MMD can distinguish distributions with
different means and different variances

\[
MMD(P, Q) = \| \mathbb{E}_{X \sim P} [\varphi(X)] - \mathbb{E}_{Y \sim Q} [\varphi(Y)] \|_H = \sqrt{(\mathbb{E}X - \mathbb{E}Y)^2 + (\mathbb{E}X^2 - \mathbb{E}Y^2)^2}
\]

• MMD can be estimated from samples using the kernel trick
  – Let \( k(x, y) = \langle \varphi(x), \varphi(y) \rangle_H \), then we have

\[
MMD^2(P, Q) = \| \mathbb{E}_{X \sim P} \varphi(X) - \mathbb{E}_{Y \sim Q} \varphi(Y) \|_H^2
= \langle \mathbb{E}_{X \sim P} \varphi(X), \mathbb{E}_{X' \sim P} \varphi(X') \rangle_H + \langle \mathbb{E}_{Y \sim Q} \varphi(Y), \mathbb{E}_{Y' \sim Q} \varphi(Y') \rangle_H
- 2 \langle \mathbb{E}_{X \sim P} \varphi(X), \mathbb{E}_{Y \sim Q} \varphi(Y) \rangle_H
= \mathbb{E}_{X, X' \sim P} k(X, X') + \mathbb{E}_{Y, Y' \sim Q} k(Y, Y') - 2 \mathbb{E}_{X \sim P, Y \sim Q} k(X, Y)
= \frac{1}{m(m-1)} \sum_{i \neq j}^m k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j}^n k(y_i, y_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j)

  – Does not require knowledge of possibly infinite-dimensional feature \( \varphi(x) \)
Maximum Mean Discrepancy

• For other features, MMD can construct stronger distances
  – Let \( \mathcal{X} = \mathbb{R} \) and \( \varphi(x) = (x, x^2) \), then MMD can distinguish distributions with different means and different variances
    \[
    \text{MMD}(P, Q) = \| \mathbb{E}_{X \sim P} [\varphi(X)] - \mathbb{E}_{Y \sim Q} [\varphi(Y)] \|_H = \sqrt{(\mathbb{E} X - \mathbb{E} Y)^2 + (\mathbb{E} X^2 - \mathbb{E} Y^2)^2}
    \]

• MMD can be estimated from samples using the kernel trick
  – Let \( k(x, y) = \langle \varphi(x), \varphi(y) \rangle_H \), then we have
    \[
    \text{MMD}^2(P, Q) = \| \mathbb{E}_{X \sim P} \varphi(X) - \mathbb{E}_{Y \sim Q} \varphi(Y) \|_H^2
    = \langle \mathbb{E}_{X \sim P} \varphi(X), \mathbb{E}_{X' \sim P} \varphi(X') \rangle_H + \langle \mathbb{E}_{Y \sim Q} \varphi(Y), \mathbb{E}_{Y' \sim Q} \varphi(Y') \rangle_H
    - 2 \langle \mathbb{E}_{X \sim P} \varphi(X), \mathbb{E}_{Y \sim Q} \varphi(Y) \rangle_H
    = \mathbb{E}_{X, X' \sim P} k(X, X') + \mathbb{E}_{Y, Y' \sim Q} k(Y, Y') - 2 \mathbb{E}_{X \sim P, Y \sim Q} k(X, Y)
    = \frac{1}{m(m-1)} \sum_{i \neq j}^m k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j}^n k(y_i, y_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j)
    \]
  – Does not require knowledge of possibly infinite-dimensional feature \( \varphi(x) \)
  – Differentiable, can be used as an objective in deep networks
Maximum Mean Discrepancy

- MMD between Gaussian and Laplacian distributions (zero mean and unit variance), computed with Gaussian kernel of $\sigma = 0.5$
Are $P$ and $Q$ different?

$P(x)$

$Q(y)$
Maximum Mean Discrepancy

Observe $X = \{x_1, \ldots, x_n\} \sim P$

Observe $Y = \{y_1, \ldots, y_n\} \sim Q$
Maximum Mean Discrepancy
Maximum Mean Discrepancy

\[ \hat{\mu}_P(v) : \text{mean embedding of } P \]

\[ \hat{\mu}_Q(v) : \text{mean embedding of } Q \]

\[ \hat{\mu}_P(v) := \frac{1}{m} \sum_{i=1}^{m} k(x_i, v) \]
Maximum Mean Discrepancy

\[ \text{MMD}^2 = \|\text{witness}(v)\|_F^2 \]

\[ = \frac{1}{n(n-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(y_i, y_j) \]

\[ - \frac{2}{n^2} \sum_{i,j} k(x_i, y_j) \]
Maximum Mean Discrepancy

- Dogs ($= P$) and fish ($= Q$) example revisited
- Each entry is one of $k(\text{dog}_i, \text{dog}_j)$, $k(\text{dog}_i, \text{fish}_j)$, or $k(\text{fish}_i, \text{fish}_j)$
The maximum mean discrepancy:

\[
MMD^2 = \frac{1}{n(n-1)} \sum_{i \neq j} k(\text{dog}_i, \text{dog}_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(\text{fish}_i, \text{fish}_j) \]

\[- \frac{2}{n^2} \sum_{i,j} k(\text{dog}_i, \text{fish}_j)\]
Determine whether GAN is good enough
Maximum Mean Discrepancy

- Empirical estimate of MMD:
  \[ \|\hat{\mu}_P - \hat{\mu}_Q\|^2_\mathcal{H} \], where
  \[
  \hat{\mu}_P := \frac{1}{m} \sum_{i=1}^{m} K(x_i, \cdot) \\
  \hat{\mu}_Q := \frac{1}{n} \sum_{i=1}^{n} K(y_i, \cdot)
  \]

- Weighted MMD
  - Control contribution of each sample to the mean embedding

  \[
  \hat{\mu}_{P,w} = \sum_{i=1}^{m} w_i K(x_i, \cdot)
  \]

  \[
  \widehat{\text{MMD}^2}(X_m, Y_n, w) := \|\hat{\mu}_{P,w} - \hat{\mu}_Q\|^2_\mathcal{H}
  \]

  \[
  = \sum_{i,j=1}^{m} w_i w_j K(x_i, x_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} K(y_i, y_j) - \frac{2}{n} \sum_{i=1}^{m} w_i \sum_{j=1}^{n} K(x_i, y_j)
  \]
Kernel Mean Matching

• Find a set of samples to represent $Q$, approximating those from $P$
  – Estimate MMD for $P$
    \[
    \hat{\mu}_{P,w} = \sum_{i=1}^{m} w_i K(x_i, \cdot)
    \]
  – Find a set of points $Y_n := \{y_i\}_{i=1}^{n} \subset \mathcal{X}$ that matches the MMD for $P$
    \[
    Y_n^* = \arg\min_{\{y_1, \ldots, y_n\}} \widehat{\text{MMD}}^2(X_m, Y_n, w)
    \]

• Consider the expression for MMD
  \[
  \widehat{\text{MMD}}^2(X_m, Y_n, w) := \left\| \hat{\mu}_{P,w} - \hat{\mu}_Q \right\|^2_{\mathcal{H}} \\
  = \sum_{i,j=1}^{m} w_i w_j K(x_i, x_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} K(y_i, y_j) - \frac{2}{n} \sum_{i=1}^{m} w_i \sum_{j=1}^{n} K(x_i, y_j)
  \]
  – Second term: encourages samples to be diverse
  – Third term: encourages distribution $Q$ to be close to $P