UNIVERSITY JEAN MONNET

Domain adaptation using Optimal Transport: application to prostate cancer mapping

Author:
Léo GAUTHERON

Supervisors:
Carole LARTIZIEN, CNRS Researcher
Ievgen REDKO, Associate Professor

A thesis submitted in partial fulfillment of the Machine Learning and Data Mining master, promotion 2015-2017

Research internship from Jan 31st 2017 to July 28th 2017 in the

CREATIS LABORATORY (Villeurbanne, France)

Contents

1 Introduction

Computer aided diagnosis systems (CAD) have been developed in the recent years to assist the medical experts in establishing their diagnosis. In the medical imaging domain, such systems can provide the location and the aggressiveness of regions of interest. This knowledge is often provided by a classification model learned with a training database. The aim being that these CAD give relevant data to doctors by detecting aggressive lesions (true positive). As data generated by the CAD are then analyzed by an expert, CAD should avoid to detect normal regions (false positive) as it will waste the time of the expert and potentially lead to diagnostic errors. Thus, a CAD efficiency is judged based on true/false positive ratio.

The effectiveness of CAD depends heavily on its training database used to learn the classification model. First, the database should contain a sufficient number of samples in the different classes of the pathology. The constitution of such database is difficult in medical imaging because it requires the medical experts to annotate a large amount of images. Then, the CAD must be deployed to classify new images coming from the same data distribution as the one from the training database. This last requirement is violated when the CAD is used on images acquired with different systems and/or different protocols. These images will indeed have different noise and texture compared to the training images. It is then probable that the use of the CAD will perform poorly on the new images. Our main objective is thus to discover **how to adapt a CAD developed with data coming from a certain imaging protocol when this CAD is to be deployed on data coming from a different imaging protocol?**

This problematic matches the one of the Domain Adaptation research field, a subfield of Transfer Learning that generalizes concepts from classic Machine Learning. In Machine Learning we want to make computers learn automatically a model from training samples such that this model is able to make predictions on new samples. For example, if we want a model that is able to detect cancer lesions in MRI images, we will learn a model from a large set of annotated images from one data distribution among which there will be images with and without lesions. The annotations are the information that was given by the expert of the domain, for example here to underline the lesions. And given a new MRI image without annotation of the same data distribution, the model might create a probability map indicating the suspicious regions. In Transfer Learning, we want to use the knowledge obtained using Machine Learning to solve a certain problem and be able to use again this knowledge to solve a similar problem. For example, we might want to use our knowledge of detecting cancer lesions in MRI to detect a different pathology such as epilepsy. The research field of Domain Adaptation is related to the one of Transfer Learning. In Domain Adaptation, we also have a classification model learned to deal with one problem. This model has been learned on a set of training samples called source domain. And we want to solve the same problem on another set of samples having a different data distribution: the target domain. There exist several cases of Domain Adaptation depending on the presence or absence of annotated data in the target domain. In semi-supervised domain adaptation, we suppose that we have a few annotated samples in the target domain. Whereas in unsupervised domain adaptation, we suppose that no labeled samples are available for the target domain. In both cases, one main assumption in domain adaptation is that the task is the same in the source and target domain.

In response of the presented problematic, we will first introduce in section 2 the medical context and data to be used in a CAD aimed to detect cancerous lesions in MRI images of prostates. After, we will study in section 3 how to adapt our CAD in the context of learning with heterogeneous data. For this, we will review in sub-section 3.1 previous work done in a similar context of fusing the images coming from different protocols [1, 2] in the case of semi-supervised domain adaptation. We will then compare these results with unsupervised domain adaptation algorithms in sub-section 3.2, some of which are based on the optimal transportation theory [13, 3, 4]. We will then present in section 5 original contributions based on the optimal transportation theory. First, we will present a method to select a subset of common features across the source and target domains in sub-section 5.1. Then we will present in sub-section 5.2 preliminary work for optimal transport aimed to automatically learn its transportation cost metric. We will then conclude in section 6.

2 Medical context

Prostate cancer is one of the most usual cancer for men. In the U.S. in 2016, it was reported the diagnosis of 180,890 new cases and the death of 26,120 men because of this cancer [5]. In France in 2016, prostate cancer is one of the top three affecting men with lung and colorectal cancers [6]. The current diagnostic techniques include rectal exam-

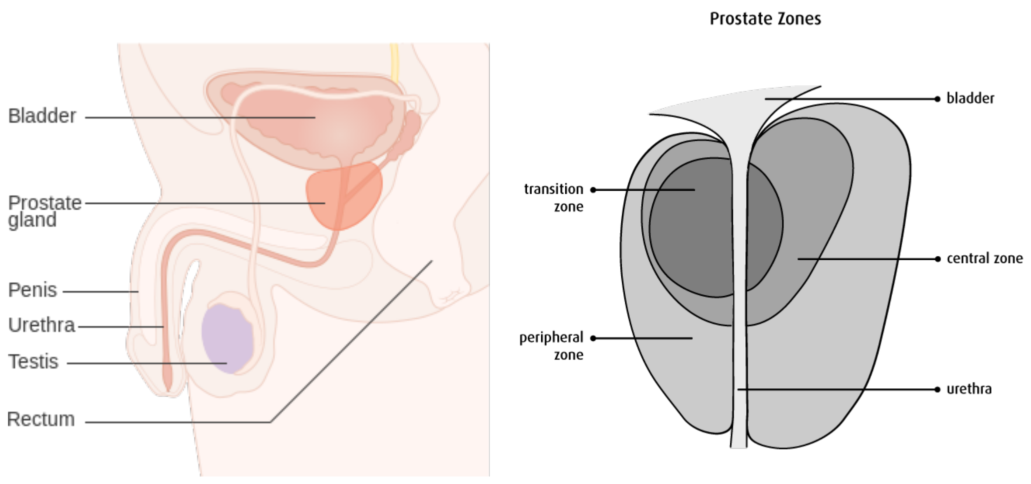


Figure 1: Left picture (source Cancer Research UK - Wikimedia) indicates the position of the prostate compared to other organs of the male reproductive system. Right picture (source www.cancer.ca) describes the different zones of the prostate. We are interested only in the peripheral zone.

ination, PSA blood measures and biopsies. But these methods have a lack of precision that may result in over/under treatment. Multi-parametric magnetic resonance imaging (MRI) is a method allowing to localize precisely cancerous regions in the prostate. In order to do this, radiologists have to analyze a large amount of MRI images on the different sequences acquired. This work is long and tedious, and the interpretation of the MRI may differ from one radiologist to another. In the recent years, there have been the development of CAD systems between hospitals and research laboratories with the aim to help interpret these large amounts of images.

In our case, there were several CAD systems developed for prostate cancer [7, 8] between the CREATIS research laboratory and the Edouard Herriot Hospital (Lyon, France). We will now describe how the data used by these CAD are obtained and used.

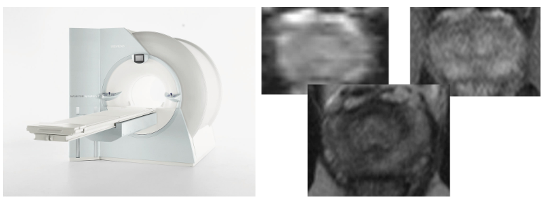


Figure 2: Left, 1.5T scanner (Symphony, Siemens Medical Systems, Erlangen, Germany). Right, example of slices produced for each sequence (bottom T2, top left ADC, top right DCE).

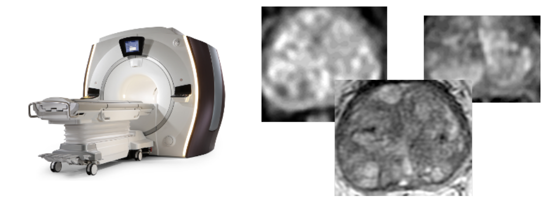


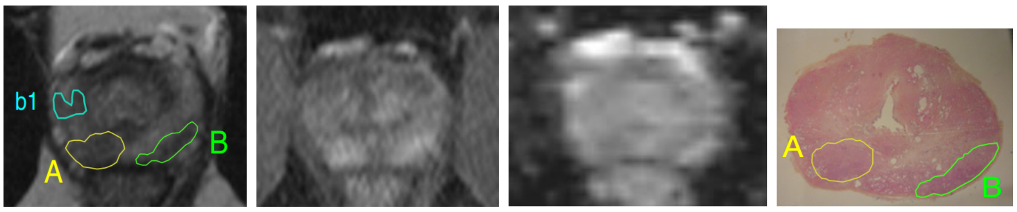
Figure 3: Left, 3T scanner (Discovery MR750 General Electric Medical Systems, USA). Right, example of slices produced for each sequence (bottom T2, top left ADC, top right DCE).

2.1 Data acquisition

In the work presented in [7], a first database of MRI images was constructed. There was 49 patients suffering from a prostate cancer that underwent an MR exam before radical prostatectomy. This exam was performed on a 1.5 Tesla clinical MR scanner (Symphony, Siemens Medical Systems, Erlangen, Germany). A total of three MR sequences were acquired: T2 (resolution 256×256), DWI (resolution 128×88) and DCE (resolution 448×512). The last sequence DCE was acquired at several times after having injected a contrast agent to the patient. The DCE sequence presenting the highest agent concentration levels was kept. The DWI and DCE sequences were then rescaled to the resolution of the T2 sequence. Each of the three sequences is composed of 24 slices having a thickness of 3mm. At the end, we obtain one 3D map of the prostate with dimensions $256 \times 256 \times 24$ for each of the three sequences. Later, the hospital performing these exams obtained a new scanner 3 Tesla (Discovery MR750 General Electric Medical Systems, USA) offering a better resolution. The MRI exams with this new scanner produce images of resolution: T2 512×512, DWI 256×256 and DCE 256×256. Apart from the resolutions, the acquisition protocol for this new scanner is the same. With this new 3T scanner, we obtain one 3D map of the prostate of dimensions $512 \times 512 \times 24$ for each of the three sequences. We dispose of the data of 41 patients that underwent the exam on this 3T scanner.

To create a CAD, not only a set of samples is needed, but also the annotations of these samples. In our case, the prostate and its cancerous lesions were outlined in the MRI images by the radiologists using OsiriX open-source image viewing workstation (Geneva, Switzerland). This work is time consuming because the radiologists have to outline the lesions in each of the 24 slices of the 3 sequences. After the MRI exam, the patients underwent a radical prostatectomy with its results analyzed by histopathologists. These results first allowed to measure the aggressiveness of the prostate cancer lesions in term of Gleason score. It then allowed to update the outlined regions in the MRI images.

The construction of a CAD depends on a classification model learned on a training set of annotated samples. As in [8], we consider each individual voxel as one sample: we suppose that the three sequences acquired T2, ADC and DCE present different views of the same 3D map of the prostate. For the 1.5T scanner, the 3D maps obtained for the exam of one patient are of resolution $256 \times 256 \times 24$. Thus we would have



(a) T_2-weighted (b) DCE (c) ADC (d) histology slide

Figure 4: Prostate MRI presenting cancer lesions (source [7]). (a) T2 sequence where cancer lesions (A and B) are outlined along with a suspicious region (b1), (b) and (c) the corresponding DCE and ADC sequences, (d) corresponding slice obtained after radical prostatectomy and analyzed by histopathologists.

1,572,864 samples (also referred as voxels) per patient. The radiologists have outlined the peripheral region of the prostate in the slices from the 24 slices where they were visible. Here, we use only the voxels belonging to these outlined regions. Inside the outlined regions of the prostate, the cancerous lesions were also outlined. Each voxel is assigned the class label corresponding to the region on which it belongs. The number of voxels available in each class for the images coming from both scanners is shown in table 1. In this work, we will focus on binary classification models. Thus, we will consider that voxels having a Gleason ≥ 7 are cancer, and the others are non cancer.

Table 1: Repartition of voxels belonging to the peripheral zone of the prostate among the different classes of cancer aggressiveness (higher Gleason score indicates more aggressive lesions). Voxels from 1.5T MRI come from 49 patients. Voxels from 3T MRI come from 41 patients. Later, we consider each individual voxel as a training sample having a binary label: Cancer $\Longleftrightarrow$ +1 (Gleason ≥ 7) or Non Cancer $\Longleftrightarrow$ -1.

Class	#voxels 1.5T	%	#Voxels 3T	%
Not Cancer	323,551	77%	786,964	80%
Suspect	31,924	8%	48,604	5%
Gleason 6	7,747	2%	10,988	1%
Gleason 7	33,507	8%	97,099	10%
Gleason 8	15,407	4%	12,372	1%
Gleason 9	7,212	2%	31,369	3%
Total	419,348		987,396	

2.2 Features description

To be able to learn a classification model for our CAD, we need our annotated samples to be described in a way allowing us to correctly discriminate them with regard to our task: to determine if a voxel is cancer or non cancer. After the MR exams, the scanners provide us with Dicom files containing the gray intensity of each voxel. From these intensities, we then extract a set of 115 features (full list in appendix A). Even though the feature extraction is out of scope of this work, we refer the interested reader to [9] to understand how our features were extracted.

The 115 features extracted were designed for MR images having a resolution of 256×256 (scanner 1.5T). Because of this, some of them present large differences between 1.5T data and 3T data. Some features on 3T data give always the same value, as can be seen in figure 5. Others were even doubloons. Therefore, we removed a set of features (full list in appendix A) from our initial set of 115 to reduce it to 95 features used. For the rest of this work, each voxel is then described by a set of 95 features.

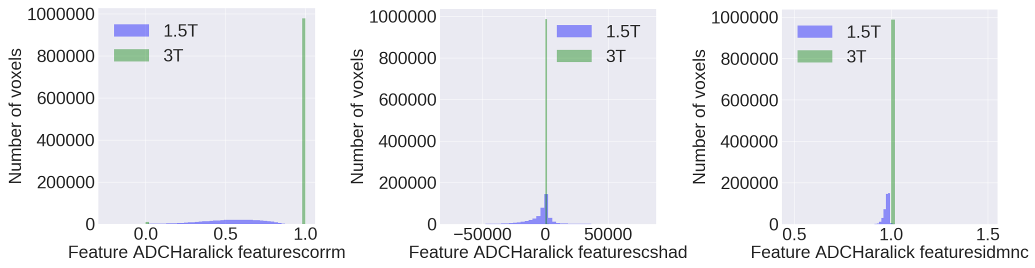


Figure 5: Some features designed for the 1.5T images do not work well for 3T images data because they give always the same value. All these features are removed.

Because of differences in the acquisition system between 1.5T and 3T data, some of these features are more or less different across the two domains, as can be seen in figure 6.

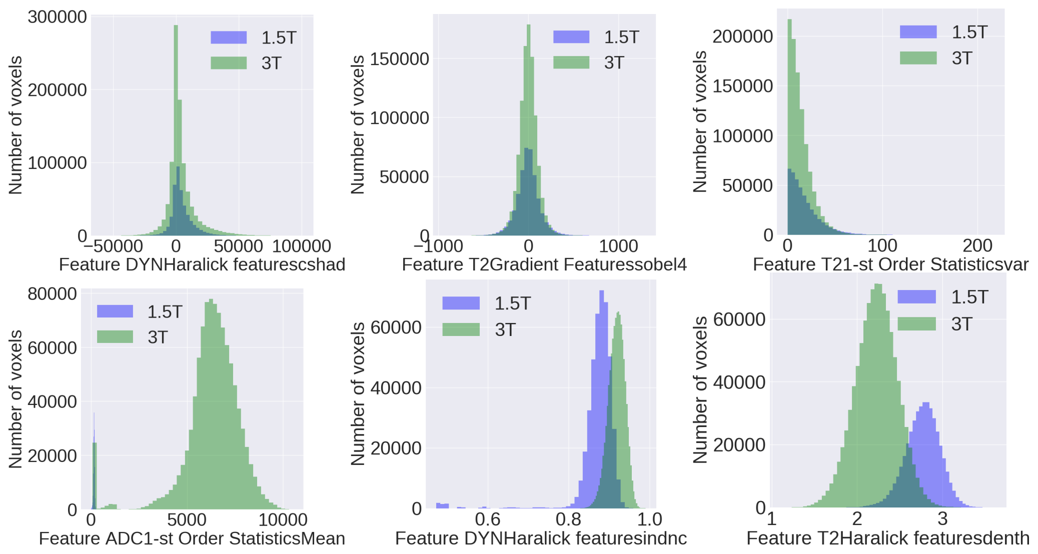


Figure 6: Example of features where the distributions are similar between the two domains 1.5T and 3T (first row) and having a shift (second row).

This observed shift between our two domains 1.5T and 3T motivates this work. In previous work on this subject, a couple of efficient CAD systems for the prostate cancer mapping were proposed and built on data that we are using. Some were trained on the 1.5T data and deployed for the 1.5T data [7, 8]. A more recent CAD was trained on the concatenation of the 1.5T and 3T data, and deployed for the 3T data [2]. Our aim is somewhat different and consists in creating a CAD trained on the 1.5T data alone, and to be deployed on the 3T data. We now propose solutions to this problem based on the domain adaptation theory.

3 Domain adaptation for prostate cancer mapping

In unsupervised domain adaptation, it is often assumed in the literature that both source and target samples have no label available to perform the adaptation. Here we relax this assumption by supposing that we dispose of the labels of all the available source samples and that the supervision of the problem depends only on the availability of labels in the target domain. Thus, in unsupervised domain adaptation, we have access to labeled source samples $\boldsymbol{S} = \{\boldsymbol{x}_i^S \in \mathbb{R}^F\}_{i=1}^n$ with labels $\{y_i^S\}_{i=1}^n$ and unlabeled target samples $\boldsymbol{T} = \{\boldsymbol{x}_i^T \in \mathbb{R}^F\}_{i=1}^m$. This scenario can be also extended to semi-supervised setting where we suppose that we know the labels of a subset $\boldsymbol{T_c}$ of the target samples: $\{y_i^T\}_{i\in\boldsymbol{T_c}}$. In both cases, the aim is to predict the unknown target labels. Unlike the classical machine learning setting, we suppose here that the marginal distributions of $\boldsymbol{S}$ and $\boldsymbol{T}$ are different. Because of this, using a classification model trained on $\boldsymbol{S}$ doesn't guarantee to obtain good performances on $\boldsymbol{T}$ if the divergence between $\boldsymbol{S}$ and $\boldsymbol{T}$ is not minimized. We can then consider to adapt $\boldsymbol{S}$ to $\boldsymbol{T}$. This adaptation method should allow us to learn a model from $\boldsymbol{S}$ such that it gives the best performances possible on $\boldsymbol{T}$.

In the literature, we distinguish three kinds of domain adaptation: (1) the reweighting methods that weight the source samples such that they become the closest from the target samples; (2) the methods aiming to find or build a shared representation space across the two domains; (3) the iterative methods that adjust iteratively the model learned on the source samples to the target samples.

3.1 Semi-supervised domain adaptation

We now present in this sub-section the different adaptations methods that were used in a previous work [2] on which we are basing our work. All these methods suppose that we dispose of annotations for some target samples $\boldsymbol{T_c}$.

Source Scaling (SS_SC): A reweighting method first described in [1] where the source samples are rescaled linearly to the target ones. Suppose that we have only two possible annotations: +1 (positive) and -1 (negative). First, we split the set of source samples $\boldsymbol{S}$ in the set of positive samples $\boldsymbol{S}^p$ and negative samples $\boldsymbol{S}^n$. We similarly split $\boldsymbol{T_c}$ in $\boldsymbol{T_c^p}$ and $\boldsymbol{T_c^n}$. We then compute the vector of means for each feature for these 4 set: $\mu(\boldsymbol{S}^p)$, $\mu(\boldsymbol{S}^n)$, $\mu(\boldsymbol{T_c^p})$ and $\mu(\boldsymbol{T_c^n})$. Each source positive/negative sample is then rescaled by a ratio of the mean between target and source:

$$\boldsymbol{S_a^p} = \boldsymbol{S}^p \times \frac{\mu(\boldsymbol{T_c^p})}{\mu(\boldsymbol{S}^p)} \text{ and } \boldsymbol{S_a^n} = \boldsymbol{S}^n \times \frac{\mu(\boldsymbol{T_c^n})}{\mu(\boldsymbol{S}^n)} \text{ and } \boldsymbol{S_a} = \boldsymbol{S_a^p} \cup \boldsymbol{S_a^n} \tag{1}$$

We can then use $\boldsymbol{S_a}$ alone or $\boldsymbol{S_a} \cup \boldsymbol{T_c}$ to learn a classifier to be deployed on the target distribution $\boldsymbol{T}$.

Enhanced Source Scaling (SS_ESC): Method described in [2] where, like in Source Scaling, we divide $\boldsymbol{S}$ in $\boldsymbol{S}^p$ and $\boldsymbol{S}^n$ and $\boldsymbol{T_c}$ in $\boldsymbol{T_c^p}$ and $\boldsymbol{T_c^n}$. First we compute the subspaces associated the these 4 sets. They are noted X_S^p, X_S^n, X_T^p and X_T^n. One subspace of a matrix of size $n \times f$ is a matrix $f \times f$ where the i^{th} column is the eigen vector having the i^{th} highest variance. The aim here is to find the matrix A such that we minimize the following equation

$$\min_A ||A||_F^2 + ||X_S^p A - X_T^p||_F^2 + ||X_S^n A - X_T^n||_F^2 \tag{2}$$

The author find that there exist a closed form solution for A defined by

$$A = (-X_S^{p'}P + X_S^{p'}X_S^p - X_S^{n'}N + X_S^{n'}X_S^n) \times (I + X_S^{p'}X_S^p + X_S^{n'}X_S^n)^{-1} \tag{3}$$

with $P = X_T^p - X_S^p$ and $N = X_T^n - X_S^n$. After having computed A, all source samples are projected in the source subspaces aligned with A:

$$\boldsymbol{S_a^p} = \boldsymbol{S}^p X_S^p A \text{ and } \boldsymbol{S_a^n} = \boldsymbol{S}^n X_S^n A \text{ and } \boldsymbol{S_a} = \boldsymbol{S_a^p} \cup \boldsymbol{S_a^n} \tag{4}$$

Similarly, we project the target samples in the source subspaces (not the target subspaces) to obtain $\boldsymbol{T}_{ca}$. We can then use $\boldsymbol{S_a}$ alone or $\boldsymbol{S_a} \cup \boldsymbol{T}_{ca}$ to learn an SVM classifier to be deployed on the target distribution $\boldsymbol{T}$. Given a test sample $x \in T$, their is a specific way for this method to determine its label. x is projected in $X_S^p A$ to obtain x_{ap} and in $X_S^n A$ to obtain x_{an}. The SVM classifier is then used to compute the distance of the two points to the learned hyperplane. This is a signed distance with a positive distance indicating a positive label, and a negative distance indicating a negative label. The signed distance of the point x to the hyperplane is then assigned to the weighted sum of the two distances. From this distance is determined the label of x.

Metric Learning (SS_ML): This method described in [2] is based on the Mahalanobis distance

$$d_M(x,x') = \sqrt{(x-x')M(x-x')} \tag{5}$$

where the aim is to learn the matrix M, here of size $m \times m$ with $m = |\boldsymbol{S}| + |\boldsymbol{T_c}|$ (note that if M is the identity matrix, we obtain the traditional euclidean distance). Here we want to learn a matrix L of size $p \times m$ with $p \ll m$ and $L'L = M$. If we replace M by $L'L$ in the Mahalanobis distance, we obtain

$$d_L(x,x') = \sqrt{(Lx - Lx')(Lx - Lx')} \tag{6}$$

which corresponds to the euclidean distance of the points in the projected space defined by L. The computation of L takes into account 4 constraints:

- For the set Sim of pairs of samples with same labels (pooled from source and target), we want to minimize the sum of their pairwise distances: $S = \sum_{(x_i,x_j) \in Sim}(x_i - x_j)(x_i - x_j)'$ multiplied by a parameter to tune γ_s.
- For the set Dif of pairs of samples with different labels (pooled from source and target), we want to maximize the sum of their pairwise distances: $D = \sum_{(x_i,x_j) \in Dif}(x_i - x_j)(x_i - x_j)'$ multiplied by a parameter to tune γ_d.
- A regularization term B using the K nearest neighbors of the samples. This is to penalize metrics changing completely the shape of the data. K is another parameter to tune. $B = (\boldsymbol{S} \cup \boldsymbol{T_c})'(diag(sum(W)) - W)(\boldsymbol{S} \cup \boldsymbol{T_c})$ with W the matrix filled with 0 and storing at (i,j) a value of 1 if sample j (from $\boldsymbol{S} \cup \boldsymbol{T_c}$) is among the K nearest neighbors of sample i, or i among the K nearest of j (symmetric matrix).
- A dimension reduction by computing the subspace matrix V composed of the d highest eigen vectors of the matrix $\boldsymbol{S} \cup \boldsymbol{T_c}$. d is thus another parameter to tune.

Consequently, γ_s, γ_d, K and d are the hyper-parameters that should be tuned. At the end, L can be calculated using a closed form solution depending on the 4 constraints presented:

$$L = 2V(B + B' + \gamma_s(S + S') - \gamma_d(D + D') + 2I)^{-1} \tag{7}$$

Finally, the source and target samples are projected in the subspace defined by L

$$\boldsymbol{S_a} = \boldsymbol{S}L \text{ and } \boldsymbol{T}_{ca} = \boldsymbol{T_c}L \text{ and } \boldsymbol{T_a} = \boldsymbol{T}L \tag{8}$$

We can then use $\boldsymbol{S_a}$ alone or $\boldsymbol{S_a} \cup \boldsymbol{T}_{ca}$ to learn a classifier to be deployed on $\boldsymbol{T_a}$. An important drawback of this method consists in a high number of parameters to tune.

3.2 Unsupervised domain adaptation

We will now present the unsupervised domain adaptation methods considered in this work. Here, we suppose that we have no label information in the target domain.

Source Scaling (US_SC): This method proposed by [1] for the semi-supervised setting is adapted to become unsupervised. We note $\mu(\boldsymbol{S})$, $\mu(\boldsymbol{T})$, $\sigma(\boldsymbol{S})$ and $\sigma(\boldsymbol{T})$ the vectors (for each feature) of means and standard deviations of source and target samples. Then the source adapted data are given by

$$\boldsymbol{S_a} = \frac{\boldsymbol{S} - \mu(\boldsymbol{S})}{\sigma(\boldsymbol{S})} \times \sigma(\boldsymbol{T}) + \mu(\boldsymbol{T}) \tag{9}$$

We can then use $\boldsymbol{S_a}$ to learn a classifier to be deployed on $\boldsymbol{T}$. The advantage of this method is its simplicity and linear complexity in the number of samples and features, but it may fail to capture a non linear transformation between source and target domain.

Subspace Alignment (US_SA): This unsupervised adaptation method [12] aims to project the source and target samples in two subspaces spanned by their principal components so that the divergence between the two domains is minimized. First we choose a number of dimensions d to compute the subspaces. These subspaces are for source $X_s \in \mathbb{R}^{F \times d}$ and for target $X_t \in \mathbb{R}^{F \times d}$. They are composed of the d highest eigen vectors having the highest variance. X_s is then aligned to X_t with the matrix M that minimizes the following objective function:

$$F(M) = ||X_s M - X_t||_F^2 \tag{10}$$

which admit this closed form solution:

$$M = X_s' X_t \tag{11}$$

The source and target samples are then projected in their respective subspaces:

$$\boldsymbol{S_a} = \boldsymbol{S} X_s M \text{ and } \boldsymbol{T_a} = \boldsymbol{T} X_t \tag{12}$$

We can then use $\boldsymbol{S_a}$ to learn a classifier to be deployed on $\boldsymbol{T_a}$.

Optimal Transport: The theory of optimal transport has been introduced by Gaspard Monge in the 18^{th} century and was recently revisited by Cédric Villani [13]. This theory gives a mathematically founded tool to align arbitrary probabilistic distributions in an optimal way.

In the discrete case, optimal transport aims to find a coupling matrix γ of two distributions defined as a joint distribution on $\boldsymbol{S} \times \boldsymbol{T}$ with empirical marginals $\hat{\mu}_S$ and $\hat{\mu}_T$ such that for all $x \in \boldsymbol{S}, y \in \boldsymbol{T}$, we minimize the transport cost from $\hat{\mu}_S$ to $\hat{\mu}_T$ with regard to a transportation cost function $c : \boldsymbol{S} \times \boldsymbol{T} \to \mathbb{R}^+$, i.e.:

$$\gamma_0 = \underset{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)}{\arg\min} \langle \gamma, C \rangle_F \tag{13}$$

where $\langle .,. \rangle_F$ is Frobenius matrix product and $C_{ij} = c(x_i^S, x_j^T)$. The constraint $\Pi(\hat{\mu}_S, \hat{\mu}_T) = \{\gamma \in \mathbb{R}_+^{N_S \times N_T} | \gamma \mathbf{1} = \hat{\mu}_S, \gamma^T \mathbf{1} = \hat{\mu}_T\}$ means that, by summing in γ the values in one row for each row, we obtain back the vector $\hat{\mu}_S$. Similarly by summing the values in one column for each column, we obtain back the vector $\hat{\mu}_T$. The original formulation of optimal transport (abbreviated **US_OT**) is a Linear Programing problem that do not scale well because of its computational complexity. This issue has been solved by [3] who proposed to add to the equation 13 a regularization on the entropy of γ:

$$\gamma_0 = \underset{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)}{\arg\min} \langle \gamma, C \rangle_F - \frac{1}{\lambda} E(\gamma) \tag{14}$$

with $E(\gamma) = -\sum_{ij} \gamma_{ij} \log \gamma_{ij}$ the entropy regularization on γ. This regularized optimal transport (abbreviated **US_OT2**) authorize the source samples to be transported more or less uniformly with regard to a regularization parameter λ to tune. This also allows this variant of the optimal transport to be optimized efficiently with the Sinkhorn-Knopp algorithm [16].

The use of optimal transport for domain adaptation has been studied for the first time in [4]. In this paper, the transport cost matrix C is defined as the pairwise squared euclidean distance between source and target samples $(x^S, x^T) \in \boldsymbol{S} \times \boldsymbol{T}$. The empirical marginals $\hat{\mu}_S$ and $\hat{\mu}_T$ are taken as uniform distributions summing to one, meaning that each source/target sample has the same weight. In this work, the authors present a new variant of optimal transport (abbreviated **US_OT3**) based on equation 14 by adding a class regularization $\ell_{0.5}\ell_1$:

$$\gamma_0 = \underset{\gamma \in \Pi(\hat{\mu}_S, \hat{\mu}_T)}{\arg\min} \langle \gamma, C \rangle_F - \frac{1}{\lambda} E(\gamma) + \eta \Omega(\gamma) \tag{15}$$

where $\Omega(\gamma) = \sum_j \sum_{\mathcal{L}} \|\gamma(I_{\mathcal{L}}, j)\|_1^{0.5}$ is the class regularization that prevent more or less (based on parameter η to tune) the source samples having a different label to be transported to the same target sample. $I_{\mathcal{L}}$ represents the list of sample indexes in $\boldsymbol{S}$ with label $\mathcal{L}$, and j goes through the sample indexes in $\boldsymbol{T}$.

After having found the optimal coupling matrix γ_0 with one of the three variants 131415, the authors of [4] propose to transport the source samples on the target ones by solving this equation for each source sample:

$$\hat{x}_i^S = \underset{x \in R}{\arg\min} \sum_j \gamma_o(i,j) c(x, x_j^T) \tag{16}$$

After that, they obtain that the solution of the equation 16 with the squared euclidean distance can be computed directly for all source samples at once. This is done with the following equation:

$$\boldsymbol{S_a} = \mathrm{diag}((\gamma_o \mathbf{1})^{-1}) \gamma_o \boldsymbol{T} \tag{17}$$

Note that, when the marginals $\hat{\mu}_S$ and $\hat{\mu}_T$ are uniform (in practice this is always the case for us), the equation 17 is simplified to

$$\boldsymbol{S_a} = N_s \gamma_0 \boldsymbol{T} \tag{18}$$

With this computation, each source samples x^S is represented as the weighted barycenter of the target samples with which it has the highest values in γ_0.

We propose a toy example in figure 7 to compare the different variants of optimal transport that are studied. We see that the classical optimal transport will associates only one target sample to each source sample. Using the variant with the regularization on the entropy, each source sample is associated to its closest target samples. By adding the class regularization, we prevent to make a coupling between samples of different classes with the same target sample. However, this do not prevent to associate a source sample for one class to a target sample of another class (because this would be a semi-supervised method requiring labels from the target domain in addition to labels in the source domain).

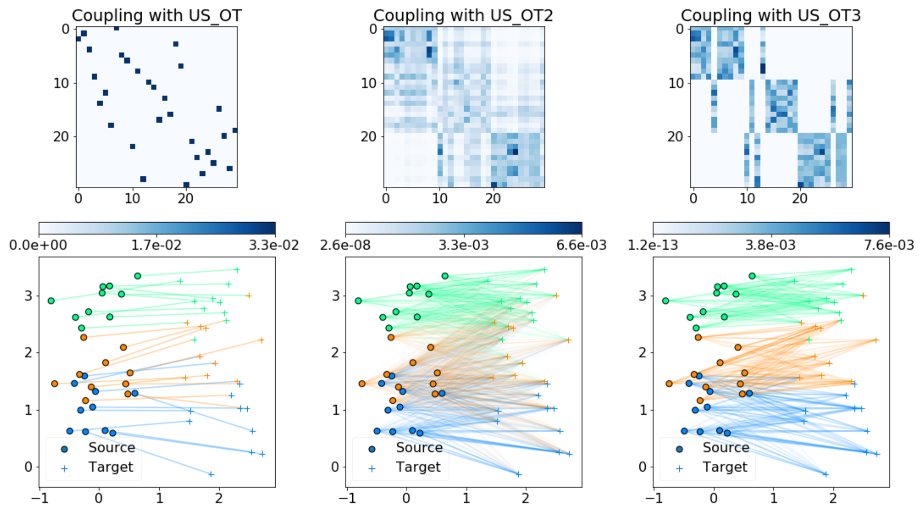


Figure 7: Comparison of the 3 variants of optimal transport studied. On the left, original optimal transport, in the middle the variation with regularization on entropy [3] ($\lambda = 1$), on the right the version with class regularization [4] ($\lambda = 1$ and $\eta = 1$). First row shows γ_0 with highest coupling values seen as darkest blue. Second row shows the source and target points composed of 3 different classes in 3 different colors (generated randomly around (0,1),(0,2),(0,3),(2,1),(2,2),(2,3)). The coupling between source and target points are shown as segments with strongest color indicating higher coupling values.

4 Experiments

After having described our medical data in section 2, and after having presented the different adaptation methods considered in sub-sections 3.1 and 3.2, we now present how these methods are used on our medical data, and how well they perform.

4.1 Methodology

Samples selection: As shown in table 1, our source $\boldsymbol{S}$ and target $\boldsymbol{T}$ domains are composed of nearly 400,000 and 1,000,000 samples. Some of our adaptation methods require to compute pairwise matrices between $\boldsymbol{S}$ and $\boldsymbol{T}$. That would require to store $4 \times 10^5 \times 1 \times 10^6 = 4 \times 10^{11}$ float numbers (4 bytes) in memory. This gives 1.6×10^{12} bytes corresponding to around 1490GB of memory! We executed our experiments in a computer equipped with 16GB of memory, thus we couldn't store these matrices in our memory. We approached this problem by selecting much smaller subsets of voxels of sizes 1500 for $\boldsymbol{S}$ and $\boldsymbol{T}$.

For the source domain $\boldsymbol{S}$, we suppose that we know the label of each sample (among the 6 classes which are: Not Cancer, Suspect, Gleason 6, Gleason 7, Gleason 8 and Gleason 9). We choose to generate a subset of size 1500 by balancing the samples of the different classes: we selected randomly 500 samples of the majority class Not Cancer, and then 200 samples for the 5 other classes. This random sample selection for one class is realized uniformly from the patients presenting sample of this class among our 49 source patients. These 1500 voxels are considered as our source domain $\boldsymbol{S_c}$.

For the target domain $\boldsymbol{T}$, we consider three subsets. Two noted $\boldsymbol{T_3}$ and $\boldsymbol{T_c}$ where we know the labels of the samples (for semi-supervised and supervised learning) and one noted $\boldsymbol{T_u}$ where we don't know them (for unsupervised learning). For $\boldsymbol{T_3}$, we selected 3 of the 41 target patients that present samples of the 6 classes. We then selected among these 3 patients the same number of voxels in each class in the same way as for our subset $\boldsymbol{S_c}$. The subset $\boldsymbol{T_c}$ is generated similarly as $\boldsymbol{T_3}$ but by taking the 1500 voxels from all the 41 target patients (subset used to learn a baseline method on target). For $\boldsymbol{T_u}$, we used an original sample selection approach based on the algorithm **US_OT** presented in sub-section 3.2. We applied the algorithm **US_OT** between the 1500 samples of $\boldsymbol{S_c}$ and 50,000 randomly sampled voxels uniformly among the 41 target patients noted $\boldsymbol{T}_{50k}$. At the end of the computation, we obtain the optimal coupling matrix γ of dimensions $1500 \times 50,000$. For each source sample in $\boldsymbol{S_c}$, we then select the target sample with which the value in the coupling matrix is maximized:

$$\boldsymbol{T_u} = \{x_j \in \boldsymbol{T}_{50k} | j = \arg\max \gamma_{ij}, i \in [0, 1500[\} \tag{19}$$

We then obtain $\boldsymbol{T_u}$ containing 1500 voxels with a sampling close to the one performed on $\boldsymbol{S_c}$ but without using any label information from the target domain.

Measure of performance used: During the development of a CAD system, we need to be able to evaluate it to see how well it performs. In the medical domain, we often evaluate such systems with the Area Under the (Receiver Operating Characteristic) Curve (abbreviated AUC). To compute the AUC, we need our classification model to return a confidence score of a given instance to belong to a given class. Here we use probabilities (between 0 and 1) with a probability close to 1 indicating a high confidence to be a cancer voxel. The computation of the AUC is described in figure 8. The AUC obtained is between 0 and 1, with higher AUC indicating better model. We also consider two other measures of performance based on the probability scores assigned to the voxels. These measures are the average probability to be a cancer voxel assigned to the cancer voxels (PC) and non cancer voxels (PNC). Both PC and PNC are between 0 and 1 as they are average of probabilities. A model can be seen as efficient with regard to these two measures if there is a large gap between PC and PNC: we want our model to give a large (respectively small) probability of being cancer to the cancer (respectively non cancer) voxels.

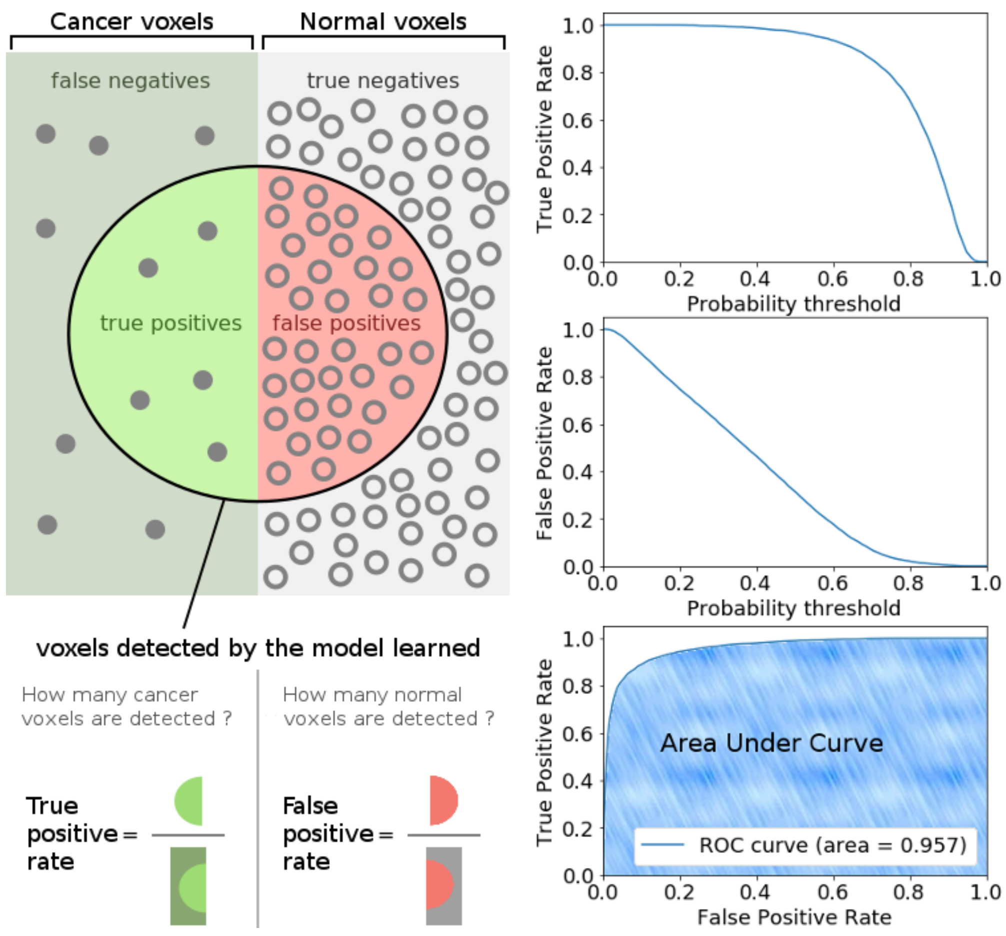


Figure 8: Graphical representation of how is computed the AUC. The classification model (Linear SVM) learned on our training voxels having annotation Cancer (+1) and non cancer (-1) assign to each testing voxel its probability to be cancer (using Platt [17] scaling). We define a set of 100 evenly spaced thresholds between 0 and 1. For one threshold t, we assign the label cancer (+1) to the testing voxels with a probability $\geq t$ and non cancer (-1) to the others testing voxels. We compute the True Positive Rate and False Positive Rate with the assigned labels. By repeating this for each threshold considered, we can plot the TPR in function of FPR, which gives the ROC curve. Finally we compute the area under this curve.

Leave One Patient Out: We adopt the LOPO strategy to evaluate our model. This technique is used when there is a small number of patients available. Suppose P is our set of patients. For each $p \in P$, we learn a model m on the voxels belonging to the patients in $P \setminus \{p\}$. Then we use m to compute the confidence scores S of the voxels belonging to patient p. After, we compute the AUC for p with S and the real labels of p. Finally, we return the mean of the AUC computed for each patient.

The AUC can be computed only on patients presenting labels from the two classes: Cancer (+1) and Not Cancer (-1), otherwise there would be a division by 0 in the computation of the TPR or FPR. We are interested only in detecting cancer in the peripheral zone of the prostate, but some patients had a cancer in another region. Because of this, those latter patients do not present cancer voxels in our data. Thus, we compute the AUC only for the patients having cancer voxels and the mean AUC returned is only over their AUC. For the source domain, this represents 36 patients among the 49 initial. For the target domain, this represents 32 patients among the 41 initial. But even if some patients are not used to evaluate the models, we still use some of their voxels in our training subsets, allowing us to have a wider variety of samples, which is important in machine learning.

Model learning and evaluation: For all experiments, we make use of a python implementation of linear SVM (sklearn.svm.LinearSVC) to learn our models. We make use of the signed distances of the samples to the hyperplane learned by SVM. These distances are rescaled in the interval $[0,1]$ using the Platt scaling [17] to give us the probabilities of the voxels to be cancer voxels. For this scaling, we first learn our SVM model m. The scaling consists in fitting a logistic regression on the signed distances obtained on the training data using m, before using it to predict the probabilities on the test adapted data. The parameter C of SVM and the different parameters of the adaptation algorithms are tuned differently in the semi-supervised and the unsupervised setting.

For semi-supervised domain adaptation, we perform a grid search on the space of set of parameters. For each set of parameters, we use them to adapt $\boldsymbol{S_c}$ to $\boldsymbol{T_3}$, giving $\boldsymbol{S}_{ca}$ and $\boldsymbol{T}_{3a}$. We then keep the model m learned on $train = \boldsymbol{S}_{ca} \cup \boldsymbol{T}_{3a}$ that maximizes the mean AUC using the LOPO strategy on $\boldsymbol{T}_{3a}$. The model m is used to classify a test set composed of all the voxels v belonging to the target patients having cancer and non cancer voxels without the 3 patients P_3 selected to build $\boldsymbol{T_3}$ (this represents 29 patients): $test = \boldsymbol{T_a} \backslash \{v \in \boldsymbol{T_a} | patient(v) \in P_3\}$. m is used to evaluate each test patient individually to obtain one AUC per patient, and we compute the mean AUC at the end. This last AUC is the performance measure considered in the results. We will compare them to a baseline method without adaptation noted **Semi-supervised No Adaptation (SS_NA)**.

For unsupervised domain adaptation, we also perform a grid search to find the best set of parameters. For each set of parameters, we use them to adapt $\boldsymbol{S_c}$ to $\boldsymbol{T_u}$, giving $\boldsymbol{S}_{ca}$. We then keep the model m learned on $train = \boldsymbol{S}_{ca}$ that maximizes the mean AUC on $train$ using the LOPO strategy. Even if we can test the model m on all the test data $\boldsymbol{T_a}$, we prefer not to do it so that we can compare semi-supervised and unsupervised methods. Thus here also m is used on $test = \boldsymbol{T_a} \backslash \{v \in \boldsymbol{T_a} | patient(v) \in P_3\}$ to compute for each test patient its AUC, and to compute the mean AUC at the end. This last AUC is the performance measure presented in the results. These results will also be compared to a baseline method without adaptation noted **Unsupervised No Adaptation (US_NA)**.

The different semi-supervised and unsupervised method will be compared to another baseline where we learn directly from the target data and deploy on the target data. For this we use our subset $\boldsymbol{T_c}$ where we sampled 1500 voxels according to their class labels from the 41 patients. We adopt a similar LOPO strategy on the set of the 29 target patients on which we evaluated the two previous methods. For each patient p of them, we learn a model m on $train = \boldsymbol{T_c} \backslash \{v \in \boldsymbol{T_c} | patient(v) = p\}$. And we use m to classify all the voxels of this patient $\boldsymbol{T}_p = \{v \in \boldsymbol{T} | patient(v) = p\}$ to compute the AUC. We then return the mean AUC over the 29 patients. This baseline method is referred as **Supervised No Adaptation (S_NA)**.

4.2 Results

Semi-supervised comparison: First, we compare the different semi-supervised methods described in 3.1 between each other. As explained in sub-section 4.1, the classification model SVM is learned on $train = \boldsymbol{S}_{ca} \cup \boldsymbol{T}_{3a}$. The four methods compared are **SS_NA**, **SS_SC**, **SS_ESC** and **SS_ML**. We will in addition consider these 4 methods by learning the model only on the source adapted data ($train = \boldsymbol{S}_{ca}$): **SS_S_NA**, **SS_S_SC**, **SS_S_ESC** and **SS_S_ML**. Finally, we also consider these 4 methods learned only on the target adapted data ($train = \boldsymbol{T}_{3a}$): **SS_T_NA**, **SS_T_SC**, **SS_T_ESC** and **SS_T_ML**.

Table 2: Comparison of the 4 semi-supervised adaptation algorithms considered on our task to adapt our source 1.5T data to our 3T target data. After applying these 4 algorithms to adapt source to target, we train a linear SVM on source union target (left), on source only (middle) and on target only (right). And we report the resulting mean (over the patients) AUC, PC and PNC: Area Under (ROC) Curve, Probability (of belonging to the cancer class) given to Cancer voxels, and Probability given to Non Cancer voxels are the measure of performance given.

Method	AUC	PC	PNC	Method	AUC	PC	PNC	Method	AUC	PC	PNC
SS_NA	0.816	0.637	0.395	SS_S_NA	0.505	0.000	0.000	SS_T_NA	0.812	0.657	0.419
SS_SC	0.800	0.677	0.350	SS_S_SC	0.660	0.623	0.354	SS_T_SC	0.802	0.667	0.395
SS_ESC	0.808	0.641	0.428	SS_S_ESC	0.578	0.409	0.406	SS_T_ESC	0.647	0.537	0.529
SS_ML	0.798	0.634	0.422	SS_S_ML	0.568	0.418	0.379	SS_T_ML	0.771	0.625	0.446

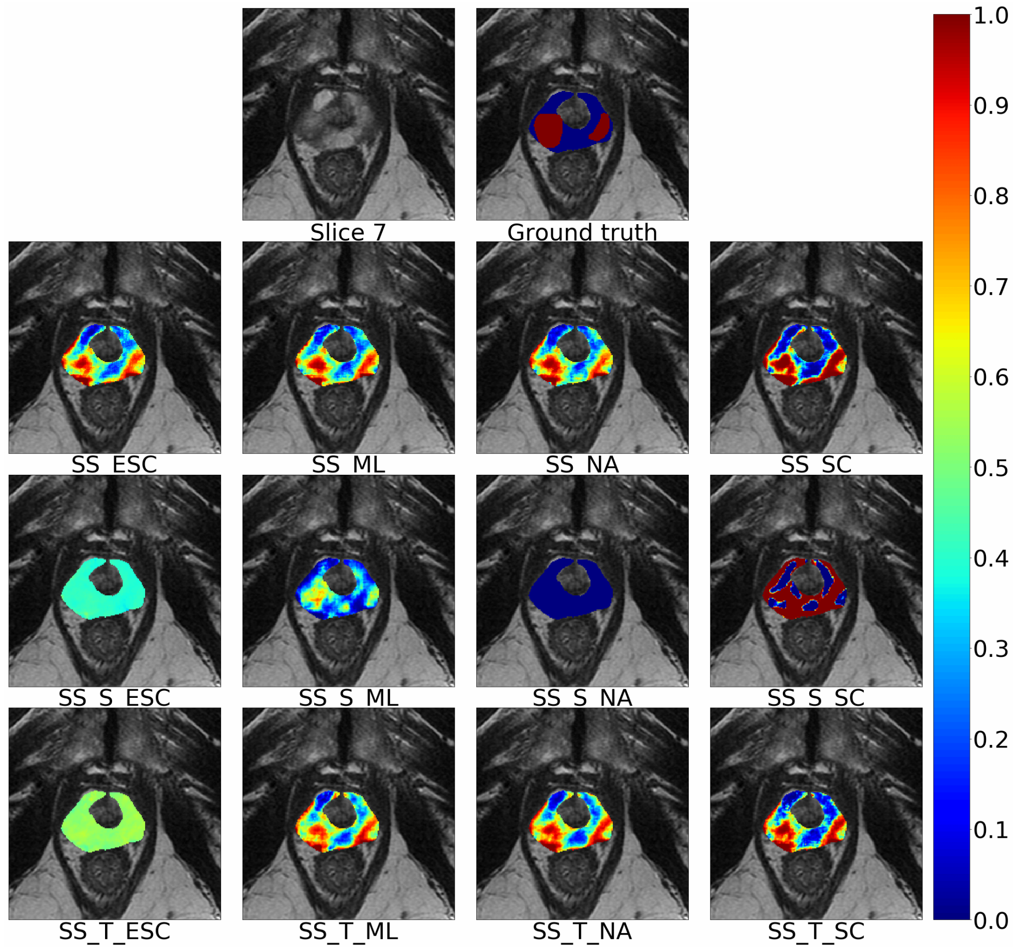


Figure 9: Figure corresponding to results presented in table 2. The first image shows the original slice of a patient having two cancerous lesions indicating in red in the ground truth image. Each other figure show the probability assigned to each voxel by the different semi-supervised adaptation methods.

The results for this experiment are shown in table 2 and figure 9. When the model is learned on the source adapted data alone we obtain an AUC of 0.505, 0.660, 0.578 and 0.568 for the 4 algorithms. But when the model is learned on the concatenation of source and target, the performances are significantly better with an AUC of respectively 0.816, 0.800, 0.808 and 0.798. It seems that using these semi-supervised methods to adapt the source data is not a good idea, unless we plan to use the source adapted data with the target data. The use of only the target data gives as expected higher results since they come from the same data distribution as the one where we evaluate the model. For the Enhanced Source Scaling method, the use of one of the two domains alone produces uniform cancer probability maps, preventing any interpretation of them by a specialist. This is the same when we learn only on the source data without adaptation. Doing this do not work at all since all the voxels are assigned a probability of 0 to be cancer. This prove the need to use an adaptation algorithm. Globally, the classification model give better performances when learned on the concatenation of the source adapted data and target adapted data than on one of the two alone.

Unsupervised comparison: Now, we compare the different unsupervised methods described in 3.2 between each other.

Table 3: Comparison of the 6 unsupervised adaptation algorithms considered on our task to adapt our source 1.5T data to our 3T target data. After applying these 6 algorithms to adapt source to target, we train a linear SVM on the source data only, and we report the resulting mean (over the patients) AUC, PC and PNC.

Method	AUC	PC	PNC
US_NA	0.505	0.000	0.000
US_SA	0.817	0.598	0.339
US_SC	0.848	0.532	0.253
US_OT	0.808	0.525	0.324
US_OT2	0.656	0.620	0.363
US_OT3	0.768	0.615	0.229

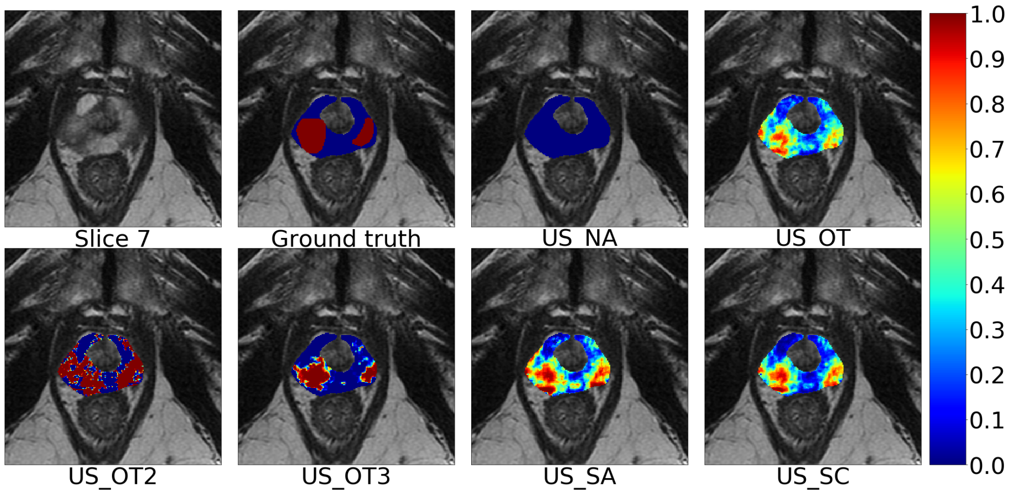


Figure 10: Figure corresponding to results presented in table 3. These results are evaluated on the same slice of the patient considered in figure 9.

As before, we see that by learning a model on the source data without adaptation, the results on the target data are always the same: all the voxels are assigned a cancer probability of 0. There is a clear shift between source and target data because the target data is so far from the decision boundary learned by the linear SVM that all target samples are always on the same side of the hyperplane, here in the negative side. But even if there is a clear shift, it seems that there is a rather easy way of getting rid of it: the Unsupervised Source Scaling method is in fact the method that gives the best mean AUC of 0.848 while also being the simplest algorithm in term of implementation and computation time. We see an interesting behavior for the algorithm OT3. It allows to make decisions with high probabilities: for Source Scaling, the PC and PNC those are 0.532 and 0.253 while they are of 0.615 and 0.229 for OT3. This gives gaps of respectively 0.279 and 0.386 between the probabilities given to the two classes. In the medical context, it may be useful to have a large separability between the different classes to help the expert to make his diagnostic faster by immediately detecting the regions of interest. As a side note, we had a paper accepted in the national conference GRETSI presenting the unsupervised domain adaptation methods and their performances.

Semi-supervised Versus Unsupervised: We now aggregate the results presented individually for semi-supervised and unsupervised methods. We also compare them with the fully supervised baseline method **S_NA** on the target data.

Table 4: Global comparison of the semi-supervised and unsupervised adaptation algorithms considered on our task to adapt our source 1.5T data to our 3T target data.

Method	AUC	PC	PNC
S_NA	0.844	0.643	0.394
SS_NA	0.816	0.637	0.395
SS_SC	0.800	0.677	0.350
SS_ESC	0.808	0.641	0.428
SS_ML	0.798	0.634	0.422
US_SA	0.817	0.598	0.339
US_SC	0.848	0.532	0.253
US_OT	0.808	0.525	0.324
US_OT2	0.656	0.620	0.363
US_OT3	0.768	0.615	0.229

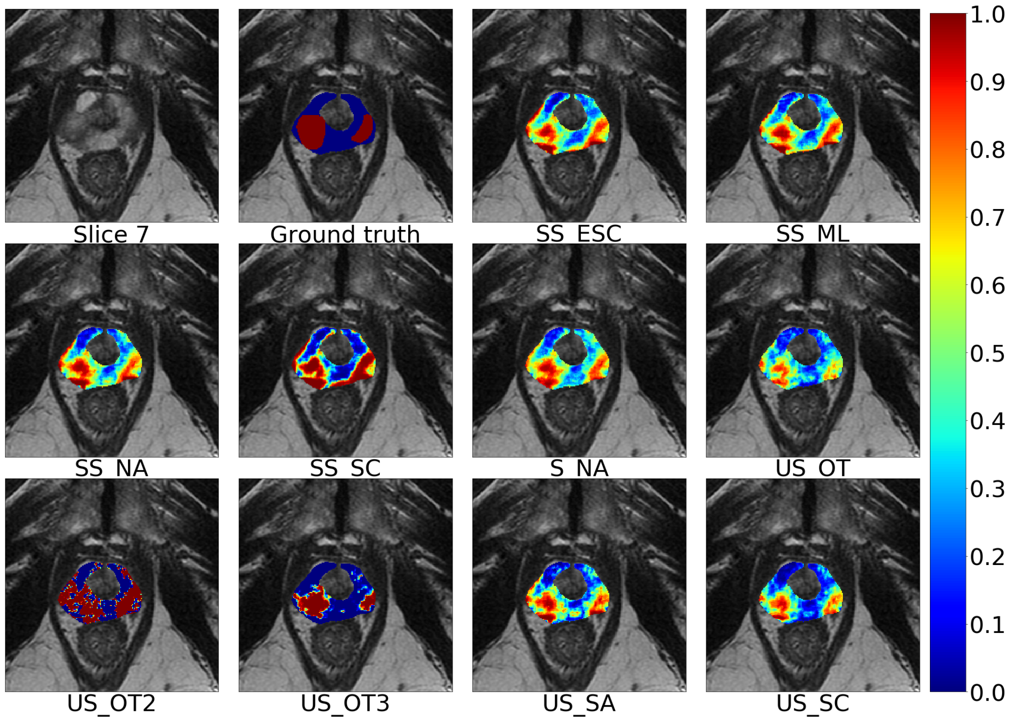


Figure 11: Figure corresponding to results presented in table 4. This figure aggregates the best results of the figures 9 and 10.

The results are shown in table 4 and figure 4. With an AUC of 0.844, the fully supervised baseline method is the second best after the method Unsupervised Source Scaling having an AUC of 0.848. This adaptation method successfully works on this adaptation task because we achieve better performances on the target domain by learning from a different data distribution. On the side of semi-supervised algorithm, the method that performs the best is Enhanced Source Scaling with an AUC of 0.808. This is as good or less good than three Unsupervised methods which are Subspace Alignment, Optimal Transport and Source Scaling. Contrarily to what could be expected, the use of labels in the target domain do not improve performances compared to the setting where we only use labels in the source domain. The concatenation of source and adapted data might still be sub-optimal if source adapted and target adapted data are still rather different. Another better way of using the target labels in this task could be to cross-validate the parameters of an Unsupervised algorithm such that it maximizes the AUC on the target data available.

Increasing number of voxels: In the previous experiments, we have used rather small training subsets: 1500 source voxels taken from the database of 419,348 voxels. And 1500 target voxels taken from the database of 987,396 voxels. Here, we propose to evaluate the performances and computation time of the unsupervised adaptation methods by changing the size of these subsets. The source subset will be generated like before 4.1 by taking $\frac{5}{15}$ of the voxels in the majority class, and $\frac{2}{15}$ of the voxels for the 5 other classes. However, the target unsupervised subset $\boldsymbol{T_u}$ (of the same size of the source one), will be generated differently. As for large number of voxels in the source subset, the US_OT algorithm used to generate $\boldsymbol{T_u}$ would ran out of memory, due to its quadratic space complexity. $\boldsymbol{T_u}$ is generated directly randomly uniformly among the 41 target patients.

The implementation of the unsupervised algorithms presented above are in Python. Optimal transport based algorithms are implemented in a library available on Github [1]. The basic implementations of the algorithms **OT2** and **OT3** are with traditional python code running on the processor (CPU) of the computer. We developed with a python library [2] an efficient implementation of these two algorithms that runs on the graphics processing unit (GPU) of the computer (in our case the GPU is an NVIDIA Titan X). We made available these GPU versions of the algorithms for everyone in the Github repository of the Optimal Transport library. For the computation time, we will in addition compare the computation time between CPU and GPU versions.

The performance results of the AUC are in table 5 while the computation times are in table 6. Concerning the AUC, for the best algorithm Source Scaling (SC), we obtain with 1500 training voxels an AUC of 0.849, and an AUC of 0.850 with 13000 voxels. The difference between the two do not seem to be significant. For other algorithms like Subspace Alignment (SA), we obtain an AUC of 0.814 with 1500 voxels and an AUC of 0.807 with 13000. In this case the performance decrease by increasing the number of training samples. By using only a subset of 100 voxels, we clearly see that the performances are worse than with 1500 because we obtain an AUC of 0.768 and of 0.693 for SC and SA respectively. From these results, it seems that the performances obtained rapidly reach a maximal value for a small number of training voxels compared to the number initially available. This could be explained by the fact that the initial space of voxels can be reduced to a much smaller subset properly reflecting the same statistical distribution.

[1] https://github.com/rflamary/POT

[2] https://github.com/cudamat/cudamat

Table 5: AUC obtained with the different unsupervised domain adaptation algorithms by increasing the number of source voxels used to adapt and to learn the linear SVM.

#voxels	SA	SC	OT	OT2	OT3
100	0.693	0.768	0.718	0.560	0.725
250	0.746	0.814	0.750	0.621	0.711
500	0.805	0.841	0.808	0.694	0.768
1000	0.790	0.839	0.812	0.668	0.770
1500	0.814	0.849	0.802	0.678	0.766
2000	0.821	0.855	0.814	0.658	0.761
3000	0.824	0.853	0.812	0.650	0.754
5000	0.809	0.851	0.815	0.649	0.752
9000	0.819	0.852	0.824	0.662	0.768
13000	0.807	0.850	0.824	0.643	0.615

Table 6: Computation time in seconds obtained with the different unsupervised domain adaptation algorithms by increasing the number of source voxels used to adapt and to learn the linear SVM. The computation time only takes into account the application of the domain adaptation algorithm.

#voxels	SA	SC	OT	OT2_CPU	OT3_CPU	OT2_GPU	OT3_GPU
100	0.03	0.02	0.01	0.02	0.27	0.03	1.30
250	0.03	0.03	0.03	0.02	0.47	0.04	1.32
500	0.03	0.03	0.10	0.06	0.95	0.05	3.06
1000	0.05	0.04	0.40	0.22	1.28	0.07	0.62
1500	0.19	0.05	1.03	0.55	2.89	0.11	0.62
2000	0.21	0.06	2.50	1.02	14.33	0.17	2.95
3000	0.23	0.05	4.77	2.51	17.84	0.27	1.27
5000	0.37	0.07	21.65	6.27	41.60	0.66	2.06
9000	0.69	0.08	64.44	19.56	110.70	2.00	4.15
13000	0.95	0.10	169.67	41.45	262.34	5.46	8.15

In term of computation time, the Source Scaling one is the faster with a computation time going from 0.05 second at 1500 voxels to 0.10 second at 13000 voxels. This was expected since the computation of this algorithm is linear in the number of samples and in the number of features. The algorithm OT2 presented in [3] had, among others, the aim to outperform the computation time of the original optimal transport OT because of the use of Sinkhorn-Knopp algorithm [16]. This increase of speed is confirmed here since the computation time of OT rapidly form a gap with the one of OT2 that give respectively computation times of 170 and 41 seconds with 13000 voxels. However, the algorithm OT3 presented in [4] is based on OT2 by using the Sinkhorn-Knopp algorithm, but the use of the class regularization adds a large increase in computation time required, making OT3 slower than the original transport with 262 seconds required for 13000 voxels. Finally, the GPU implementations allow to decrease the computation time greatly. At 13000 voxels, the algorithms OT2 and OT3 that required 41 and 262 seconds fall to respectively 5 and 8 seconds. This is a large gain in computation time. But by increasing even more the number of voxels, these GPU implementations are also expected to require large computation time due to their polynomial complexity in the number of samples.

5 Original contributions

We have seen in the previous section how to successfully use optimal transport based algorithms to adapt the data coming from one distribution to classify data from another distribution. We now present an original method using optimal transport to select a subset of common features between two data distributions in sub-section 5.1. Then we present preliminary work in sub-section 5.2 with the aim to compute the optimal coupling between two distributions while at the same time learning the transportation cost metric.

5.1 Feature selection using optimal transport

Description of the method: Until now, we have used the optimal transport between matrices $\boldsymbol{S}$ and $\boldsymbol{T}$ that were both of dimensions $n \times f$ with $n = 1500$ voxels and $f = 95$ features. The application of an optimal transport based algorithm between two such matrices produce the optimal coupling matrix γ of dimensions $n \times n$. The geometric interpretation is that, to minimize the divergence between the distribution of $\boldsymbol{S}$ and $\boldsymbol{T}$, we can associate the source samples with the target samples with which they have the highest coupling values. The idea of the proposed method is to pass from the sample representation space (matrices $n \times f$) to the feature representation space (matrices $f \times n$). Then we apply the OT2 [3] algorithm between $\boldsymbol{S}'$ and $\boldsymbol{T}'$ giving a coupling matrix γ of dimensions $f \times f$. Geometrically, we suppose that if the feature i is similar between the two domains, then the value γ_{ii} should be large. And this value should be small if the features are different.

To transport the source features on the target features, the features have to be described by a list of source and target samples. Intuitively, there should be some kind of correspondence between the sample number i describing the source features and the sample number i describing the target features. There is thus a preliminary step consisting in finding which sample will be used to describe the features in the source and target domains. For this, we propose to use the method presented in sub-section 4.1 where we apply the OT algorithm between the source and target samples before associating each source sample with the target one corresponding to the highest coupling.

Given the optimal coupling γ of dimensions $f \times f$, we sort the features into F where the feature number i in F is the one having the i^{th} highest coupling value between the two domains. Algorithm 1 summarize our proposed method. After having obtained the ordered list of features F, it is then possible to use the $d < f$ first features of F for the classification problem at hand. This method can be applied as a preprocessing before using a domain adaptation algorithm to discard the features that are completely different across the two domains. This can also be applied if no adaptation is used to select the common features between training and test data.

Algorithm 1: Unsupervised feature selection for domain adaptation

```
Input: Source labeled matrix $\boldsymbol{S}$ ($n \times f$), Target matrix $\boldsymbol{T}$ ($m \times f$)
Output: The order $F$ of the $f$ features by decreasing similarity between $\boldsymbol{S}$ and $\boldsymbol{T}$
  Generate $\boldsymbol{S_c}$ of dim $x \times f$ representing the different classes s.t. $x \leq n$ and $x \leq m$
  Generate $\boldsymbol{T_u}$ of dim $x \times f$ using equation 19
  Normalize $\boldsymbol{S_c}$ and $\boldsymbol{T_u}$ independently to have both zero mean and unit variance.
  $\gamma = OT2(\boldsymbol{S}'_{\boldsymbol{c}}, \boldsymbol{T}'_{\boldsymbol{u}})$ {Algo [3] applied on the transposed matrices}
  $F = argSortDesc(\{\gamma_{ii} | i \in [0, f[\})$ {Returns the features sorted by $\gamma_{ii}$}
```

Experiments on Office Caltech: With this algorithm, we suppose that the features available are not all useful. This is the case of datasets described with Bag of Words (BOW) methods where a large set of f predefined words is first established: each sample s is described by f features where the value of the feature i is the number of occurrences of the word i in the sample s. As an example of dataset described with BOW in domain adaptation, we will use the Office-Caltech dataset of images. In this dataset, the classification task is to assign an image with the class of the object it contains [3]. It is composed of 4 domains A, C, W and D containing respectively 958, 1123, 295 and 157 images belonging each to one of 10 different classes. We will use the SURF [18] features made available on the website of the dataset with which each image is described by a feature vector of size 800. We will also use neural network encodings that we extracted from two pretrained neural networks which are CaffeNet (based on AlexNet [19]) and GoogleNet [20]. These neural networks were trained on the large dataset of images ImageNet containing millions of images where the classification task is to classify images among 1000 classes. By default, these networks return a probability vector of size 1000. We remove the last layer of these networks and use the output of the previous layer having a size of 4096 for CaffeNet and a size of 1024 for GoogleNet. To resume, for each image of the Office-Caltech dataset, we give them as input of the pretrained neural networks, and we obtain as output a feature vector of size 4096 or 1024 describing this image.

We compare the unsupervised domain adaptation algorithms **US_NA** and **US_OT3** on the presented dataset. Our experimental protocol to evaluate the proposed method is based on the one used in [4]. We consider the 12 domain adaptation problems by taking repeatedly one of the domains A, C, W and D as source and one of the three remaining as target $\boldsymbol{T}$. For one domain adaptation problem, we randomly sample 20 images per class (8 per class if the source domain is D). This gives us 200 images (resp. 80) as our source domain $\boldsymbol{S}$. We then apply algorithm 1 with $\boldsymbol{S}$ and $\boldsymbol{T}$ to obtain the ordered list of features F by decreasing similarity between $\boldsymbol{S}$ and $\boldsymbol{T}$. For an increasing number of features d, we use the d first features of F to adapt $\boldsymbol{S}$ and $\boldsymbol{T}$ with one of the two adaptation algorithms. After, we use a 1NearestNeighbor algorithm using the source adapted data $\boldsymbol{S_a}$ as training to compute the classification accuracy on the target data $\boldsymbol{T}$. We repeat this 10 times, and we report the mean accuracy over the 10 iterations. An extract of the results without adaptation (**US_NA**) is in figure 12 (full results are in appendix in figures 151719) and for the domain adaptation algorithm **US_OT3** in figure 13 (full results in figures 161820).

[3]https://cs.stanford.edu/ jhoffman/domainadapt/

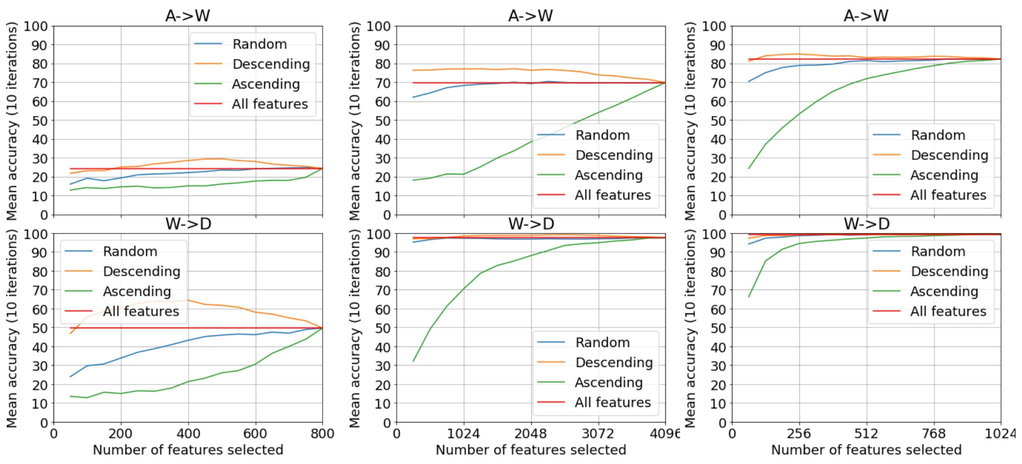


Figure 12: Result of our proposed feature selection method on 2 adaptation problems (over the 12 possible) with the algorithm **US_NA** (no adaptation). First column using surf histograms, second column using CaffeNet encoding and last column using GoogleNet encoding. Our proposed approach consists in selecting the features by Descending order of their coupling between source and target data. In addition, we plot the results by performing a selection in reverse order: Ascending, and by selecting Randomly the features.

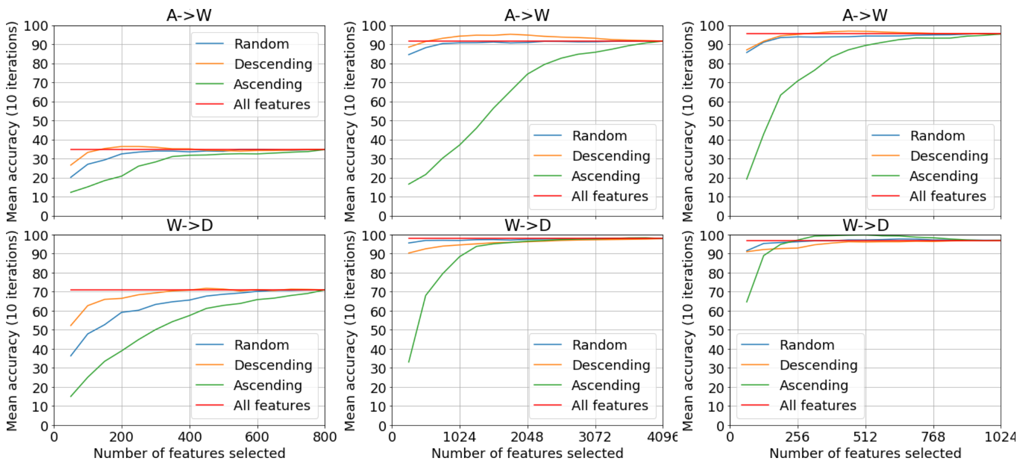


Figure 13: Same results as in figure 12 but by using the **US_OT3** domain adaptation algorithm.

From the results in 12 we see that globally, the values of the curve Descending are always higher than the values of the curve Random, itself always higher than the curve Ascending. This means that by taking first the features the more similar between the two domains (by Descending order of similarity), we achieve a better accuracy than by taking them in Ascending order. With SURF histograms for $W \rightarrow D$, we obtain with the 800 features a mean accuracy of 50%. By selecting only the 300 first features having the highest coupling, we obtain a mean accuracy of 65%. And selecting the 300 first features having the lowest coupling gives an accuracy of 18%. This selection reflects that by taking similar features between the two domains, we make the job easier for the classification model that can achieve better performances with a smaller number of features. For neural encodings, we see that the curve Random is very close to the Descending one. But there is also a large gap between the Ascending curve and the two others for a small number of features selected. Here the behavior shown is that if we take the features dissimilar between the two domains, then it is likely that the performances of the resulting classification model will be poor. By taking $\frac{1}{4}$ of the neural features randomly, we obtain on average the same performances as by taking all the features. This could be explained by the fact that the features extracted by neural networks are correlated between each other and that taking a representative subset of them is as good as taking all of them. Taking $\frac{1}{16}$ of the neural features that have the highest similarity allows to give as good if not better performances as by taking all of them. If we compare the results with (figure 13) and without (figure 12) adaptation, we obtain as expected better results by adapting the data. Globally, the results with all features (red lines) are higher with adaptation than without. We also see that with adaptation, the Ascending curve increase to its maximal value sooner than without adaptation. This can be explained by the fact that there are less features that are different across the two domains when we adapt the data. Intuitively, we can think that, the better the adaptation algorithm, the sooner the Ascending curve reach its maximal value. Because the better the adaptation algorithm, the less the number of features different across the two domains.

Experiments on our prostate data: We can also apply our proposed feature selection approach to our prostate data. Indeed, we have for each voxel a list of 95 features. As shown in sub-section 2.2, some of these features have a clear shift between our source and target domains. Because of this, a classification model learned only on the source data without adaptation completely fails to classify correctly the target data, as shown in table 3 (algorithm US_NA). We repeat the protocol presented in sub-section 4.1 in the context of our feature selection method. Here we will compare only the algorithms **US_NA**, **US_SC** and **US_OT3**.

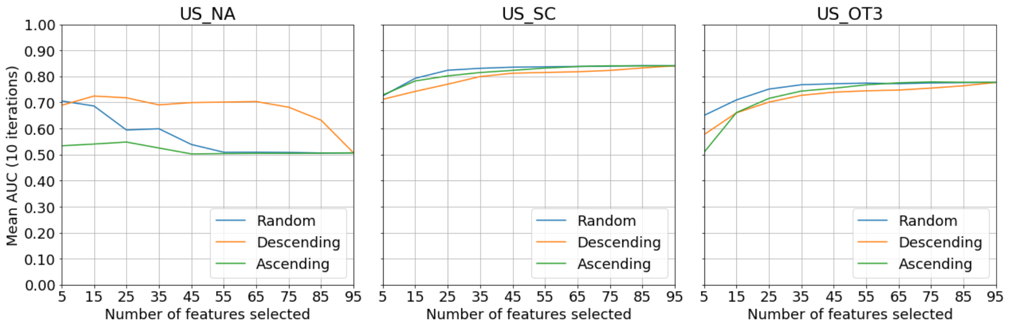


Figure 14: Results of our proposed feature selection method on the task to adapt our 1.5T prostate data to our 3T data. Each figure shows the results with one of three unsupervised adaptation algorithm presented previously

For the Office Caltech dataset, we saw that the more we take features, the higher the performances. Here however, without adaptation (algorithm US_NA), the more we take features, the smaller the AUC is, and thus the classification model becomes less and less accurate. But we observe without adaptation a similar behavior as the one presented for the Office Caltech dataset: the values of the curve Descending are always higher than the values of the curve Random, itself always higher than the curve Ascending. In fact, until 65 over 95 features with the decreasing order, the AUC is of 0.70. Then it falls sharply to 0.5. It seems that there is a small subset of features shifted between the two domains. And if we discard them, we are able to learn a model to deal with the task, even if the performances are not exceptional. When we look at the performances of the two adaptation algorithms US_SC and US_OT3, we see a different behavior. It seems that taking the features randomly gives better performances than by taking them in Ascending order of their similarity, itself giving higher performances than by taking them randomly. This behavior could be expected: when we select the features by increasing or decreasing order of their similarity between the two domains, we do not take into account how much these features contribute to correctly solve our classification task. In this case, we can think that the features allowing to give good performances are the ones in the middle. Overall, we see that the performances rapidly reach their maximum with 35 over 95 features selected. The proposed feature selection method seems to work for our handcrafted features only when no adaptation is applied later.

5.2 Optimal transport with learned transport cost metric

Given two distributions $\boldsymbol{S}$ and $\boldsymbol{T}$, an algorithm solving equation 14 will output the optimal coupling γ between $\boldsymbol{S}$ and $\boldsymbol{T}$. This coupling is computed with regard to the transportation cost matrix C. Until now, we have used in all our experiments C as the matrix of pairwise squared euclidean distances between each pair $(x^S, x^T) \in \boldsymbol{S} \times \boldsymbol{T}$. Our idea is to learn (in addition to γ) at the same time the transport cost metric C. For this, we want to learn a kernelized metric that would allow to capture non linearity between the two distributions using the kernel trick. We base this work on the idea of multiple kernel learning [11].

In multiple kernel learning, we suppose that we have a set of m predefined kernels K between our two distributions:

- Gaussian kernels $\exp(-\frac{||x_S - x_T||^2}{2\sigma^2})$ for different values of σ;
- Polynomial kernels $(x_S' x_T + 1)^d$ for different values of d.

Our preliminary proposed problem to solve is:

$$\begin{aligned} J(\gamma,\mu) = & \underset{\gamma \in \Pi(\hat{\mu}_S,\hat{\mu}_T),\mu}{\arg\min} \ \langle \gamma, C \rangle_F - \frac{1}{\lambda} E(\gamma) \\ \text{s.t.} & \\ & C_{i,j} = 1 - \sum_{t=1}^{m} \mu_t K_t(x_i^S, x_j^T) \\ & \sum_{t=1}^{m} \mu_t = 1 \text{ and } \mu_t \geq 0 \end{aligned} \tag{20}$$

where μ is the vector of size m with values summing to 1 and μ_t the weight of the predefined kernel K_t. The equation to solve is the same as the one proposed by [3] with in addition the parameter μ and two constraints depending on it. Using the Bonnans theorem [21], we know that $J2(\mu) = min_\gamma J(\gamma,\mu)$. This equation means that to solve 20, we can optimize alternatively γ and μ by fixing one of the two and optimizing the other. To optimize γ, we can employ the efficient algorithm of [3] using initially μ as the uniform vector summing to 1. To update μ, we propose to use a gradient descent algorithm.

The gradient descent requires to compute the gradient of the function $J(\mu)$. Let us now compute it:

$$\begin{aligned}
\frac{\partial J}{\partial \mu} =& \partial(\langle \gamma, \mathbf{1} - \sum_{t=1}^{m} \mu_t K_t \rangle_F - \frac{1}{\lambda} E(\gamma)) \\
=& \partial(\mathrm{tr}((\mathbf{1} - \sum_{t=1}^{m} \mu_t K_t)' \gamma)) \\
=& \mathrm{tr}(\partial((\mathbf{1} - \sum_{t=1}^{m} \mu_t K_t)' \gamma)) \\
=& \mathrm{tr}(\partial((\mathbf{1} - \sum_{t=1}^{m} \mu_t K_t)') \gamma) \\
=& \mathrm{tr}((-\partial(\sum_{t=1}^{m} \mu_t K_t)') \gamma) \\
=& \mathrm{tr}((-\left[K'_1, ..., K'_m\right]) \gamma) \\
=& \begin{pmatrix} -\mathrm{tr}(K'_1 \gamma) \\ ... \\ -\mathrm{tr}(K'_m \gamma) \end{pmatrix} \\
=& \begin{pmatrix} \langle K_1, -\gamma \rangle_F \\ ... \\ \langle K_m, -\gamma \rangle_F \end{pmatrix}
\end{aligned} \tag{21}$$

With η as the gradient step, the update $\mu = \mu - \eta \frac{\partial J}{\partial \mu}$ is computed using:

$$\mu_i = \mu_i - \eta \langle K_i, -\gamma \rangle_F \tag{22}$$

for each μ_i. $\langle K_i, -\gamma \rangle_F$ is computed as a term to term matrix product between K_i and $-\gamma$ and then by summing all the values of the resulting product matrix. After having updated the vector μ, it is very likely that it will no longer satisfy the constraint of having its values summing to 1: $\sum_{t=1}^{m} \mu_t = 1$. We then project the point μ on the nearest point x that belongs to the simplex by solving the equation:

$$\begin{aligned}
\mu =& \operatorname*{arg\,min}_{x \in \Delta^m} ||x - \mu|| \\
\text{s.t. } \Delta^m =& \{(x_1, ..., x_m) \in \mathbb{R}^m | (\sum_{i=1}^{m} x_i) = 1, x_i \geq 0\}
\end{aligned} \tag{23}$$

Having found μ, we can recompute the new transportation cost C with the new weighted sum of kernels. Then we can again optimize γ with C and then again until μ converges.

After having learned μ and γ, we want to be able to transport the source samples to the target ones, as done in [4] through equation 17. In there paper, the authors explain that this equation is valid only with C equal to the pairwise squared euclidean distance matrix. But here, we learn C as a weighted sum of kernels, and equation 17 can no longer be used. We can then solve for each source point the equation:

$$\hat{x}_i^S = \operatorname*{arg\,min}_{x \in \mathbb{R}} \sum_j \gamma(i,j) c(x, x_j^T) \tag{24}$$

This equation can be solved with specialized optimization toolkits based on the Broyden–Fletcher–Goldfarb–Shanno method (BFGS) allowing to solve non linear optimization problems without constraints. This is the case here since we project the source samples after having found γ and μ. When using BFGS algorithms, we can specify the initial point $x \in \mathbb{R}$. We propose as heuristic to use, for one source point x_i^S, the target point x_j^T with which it has the maximum coupling value in γ.

In this current version, our optimization problem do not work well. In fact, it has been found that at the end, the vector μ will give a weight of 1 to one kernel and a weight of 0 to all the others. The kernel which is assigned a weight of 1 is the kernel having the mean value closest to 0. By looking at our optimization problem $\langle\gamma, C\rangle_F$, we see that a matrix C of zeros is a trivial solution that allows to minimize the problem.

Future work: The report of this internship is made for the 30th of June while the internship ends the 28th of July, one month after. During this future month, we will explore how to constrain the transport cost matrix C to avoid to learn the trivial solution 0. The aim would be to establish what should induce a high/small transport cost. For this, we have several ideas. The first one is to use the class regularization from [4] shown in equation 15. The idea with this constraint is that, we would like the transport cost to look like in figure 7 as a matrix with blocs on the diagonal. We will also explore another idea based on the preservation of the distances between source samples before and after barycentric interpolation. With this, the idea is to add a constraint to prevent to transport a set of source samples that were separable but become inseparable after the interpolation.

6 Conclusion

The main motivation of this internship was to study and extend domain adaptation algorithms for the task of prostate cancer detection in MRI images. The algorithms proposed in the previous research in the subject were semi-supervised because they used some annotations from the target domain. We extended the algorithms in the direction of unsupervised domain adaptation that do not need any label information from the target domain to learn a classification model to be deployed on this target domain. We presented extensive empirical evaluations between the different semi-supervised and unsupervised approaches considered. Some of the unsupervised approaches are based on the optimal transportation theory, and it was shown that the use of the optimal transport method using class regularization is the one that give the sharpest distinction between cancer and non cancer tissue. However, this sharpness comes at the cost of a precision that may not always be good. We have also shown that a simple linear transformation of the source data allowed to learn a classifier that gives the best performances on the target data. Some of the results presented in this study were submitted and accepted for publication in the GRETSI 2017 conference where we will give an oral presentation in September.

Another motivation was to study how the optimal coupling matrix γ of the optimal transport could be used for domain adaptation. We presented an original unsupervised feature selection method for domain adaptation. This method allows to quantify how much a feature is shifted between the source and target domain with the use of optimal transport. It was shown that, when no adaptation is used, we can increase the performance of a classification model by discarding the features that are the least similar between the two domains.

Finally, we are currently investigating how to modify the formulation of the optimal transport. Our aim is to not only learn the optimal coupling γ, but also to learn the associated transport cost metric. For this, we want to use the idea of Multiple Kernel Learning to learn C as a weighted sum of kernels. In the future for the end this internship, we will continue the work started in this direction.

References

[1] R. Aljundi, J. Lehaire, F. Prost-Boucle, O. Rouvière and C. Lartizien. *Transfer Learning for Prostate Cancer Mapping Based on Multicentric MR Imaging Databases.* In Medical Learning Meets Medical Imaging, 74-82, 2015.

[2] R. Aljundi. *Transfer Learning for Prostate Cancer Mapping Based on Multicentric MR imaging databases.* Master's thesis, University Jean Monnet, 2015.

[3] M. Cuturi. *Sinkhorn distances: Lightspeed computation of optimal transport.* Proceedings of NIPS, 2292-2300, 2013.

[4] N. Courty, R. Flamary and D. Tuia. *Domain adaptation with regularized optimal transport.* Proceedings of ECML/PKDD, 1-16, 2014.

[5] R. L. Siegel, K. D. Miller and A. Jemal. *Cancer statistics, 2016.* CA: a cancer journal for clinicians, vol. 66, no 1, p. 7-30, 2016.

[6] Institut national du cancer. *Les cancers en France en 2016 - L'essentiel des faits et chiffres.* 2016.

[7] E. Niaf, O. Rouvière, F. Mège-Lechevallier, F. Bratan and C. Lartizien. *Computer-aided diagnosis of prostate cancer in the peripheral zone using multiparametric MRI.* Phys. Med. Biol., 57(12): 3833-3851, 2012.

[8] J. Lehaire, R. Flamary, O. Rouvière et C. Lartizien. *Computer-aided diagnostic system for prostate cancer detection and characterization combining learned dictionaries and supervised classification.* IEEE ICIP, 2251-2255, 2014.

[9] E. Niaf. *Aide au diagnostic du cancer de la prostate par IRM multi-paramétrique : une approche par classification supervisée.* PhD thesis, University Lyon 1, 2012.

[10] J. Lehaire. *Détection et caractérisation du cancer de la prostate par images IRM 1.5T multiparamétriques.* PhD thesis, University Lyon 1, 2016.

[11] J. Zhuang, J. Wang, S. Hoi and X. Lan. *Unsupervised multiple kernel learning.* Journal of Machine Learning Research-Proceedings Track, 20, 129-144, 2011.

[12] B. Fernando, A. Habrard, M. Sebban and T. Tuytelaars. *Unsupervised visual domain adaptation using subspace alignment.* Proceedings of ICCV, 2960-2967, 2013.

[13] C. Villani. *Optimal transport, old and new.* Springer Science & Business Media, 2008.

[14] S.J. Pan and Q. Yang. *A survey on transfer learning.* In IEEE Trans. Knowl. Data Eng. 22(10):1345-1359, 2010.

[15] A. van Opbroek, M. A. Ikram, M. W. Vernooij and M. de Bruijne. *Transfer learning improves supervised image segmentation across imaging protocols.* In IEEE Trans Med Imaging, 34(5):1018-30, 2015.

[16] P.A. Knight. *The Sinkhorn-Knopp algorithm: convergence and applications.* SIAM Journal on Matrix Analysis and Applications, 30 (1) pp. 261-275, 2008.

[17] J. Platt. *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.* Advances in large margin classifiers, 10(3), 61-74, 1999.

[18] H. Bay, T. Tuytelaars and L. Van Gool. *Surf: Speeded up robust features.* Computer vision–ECCV 2006, 404-417, 2006.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems, pages 1097-1105, 2012.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[21] A. Rakotomamonjy, F.R. Bach, S. Canu and Y. Grandvalet. *SimpleMKL.* Journal of Machine Learning Research, 2008, vol. 9, no Nov, p. 2491-2521.

Appendices

A Feature on voxel used

Table 7: Total list of the 115 features extracted for each voxel coming from MR images acquired on the 1.5T scanner and on the 3T scanner.

T2ROUGHT2w	T21-st Order StatisticsMean	T21-st Order StatisticsMedian
T21-st Order Statisticsstd	T21-st Order Statisticsvar	T2Haralick featuresautoc
T2Haralick featurescontr	T2Haralick featurescorrm	T2Haralick featurescoorp
T2Haralick featurescprom	T2Haralick featurescshad	T2Haralick featuresdissi
T2Haralick featuresenerg	T2Haralick featuresentro	T2Haralick featureshomom
T2Haralick featureshomop	T2Haralick featuresmaxpr	T2Haralick featuressosvh
T2Haralick featuressavgh	T2Haralick featuressvarh	T2Haralick featuressenth
T2Haralick featuresdvarh	T2Haralick featuresdenth	T2Haralick featuresinf1h
T2Haralick featuresinf2h	T2Haralick featuresindnc	T2Haralick featuresidmnc
T2Gradient Featuressobel1	T2Gradient Featuressobel2	T2Gradient Featuressobel3
T2Gradient Featuressobel4	T2Gradient Featuressobel5	T2Gradient Featureskirsch
T2Gradient Featuresgrad1	T2Gradient Featuresgrad2	ADCROUGHADC_0-800
ADC1-st Order StatisticsMean	ADC1-st Order StatisticsMedian	ADC1-st Order Statisticsstd
ADC1-st Order Statisticsvar	ADCHaralick featuresautoc	ADCHaralick featurescontr
ADCHaralick featurescorrm	ADCHaralick featurescoorp	ADCHaralick featurescprom
ADCHaralick featurescshad	ADCHaralick featuresdissi	ADCHaralick featuresenerg
ADCHaralick featuresentro	ADCHaralick featureshomom	ADCHaralick featureshomop
ADCHaralick featuresmaxpr	ADCHaralick featuressosvh	ADCHaralick featuressavgh
ADCHaralick featuressvarh	ADCHaralick featuressenth	ADCHaralick featuresdvarh
ADCHaralick featuresdenth	ADCHaralick featuresinf1h	ADCHaralick featuresinf2h
ADCHaralick featuresindnc	ADCHaralick featuresidmnc	ADCGradient Featuressobel1
ADCGradient Featuressobel2	ADCGradient Featuressobel3	ADCGradient Featuressobel4
ADCGradient Featuressobel5	ADCGradient Featureskirsch	ADCGradient Featuresgrad1
ADCGradient Featuresgrad2	DYN1-st Order StatisticsMean	DYN1-st Order StatisticsMedian
DYN1-st Order Statisticsstd	DYN1-st Order Statisticsvar	DYNHaralick featuresautoc
DYNHaralick featurescontr	DYNHaralick featurescorrm	DYNHaralick featurescoorp
DYNHaralick featurescprom	DYNHaralick featurescshad	DYNHaralick featuresdissi
DYNHaralick featuresenerg	DYNHaralick featuresentro	DYNHaralick featureshomom
DYNHaralick featureshomop	DYNHaralick featuresmaxpr	DYNHaralick featuressosvh
DYNHaralick featuressavgh	DYNHaralick featuressvarh	DYNHaralick featuressenth
DYNHaralick featuresdvarh	DYNHaralick featuresdenth	DYNHaralick featuresinf1h
DYNHaralick featuresinf2h	DYNHaralick featuresindnc	DYNHaralick featuresidmnc
DYNGradient Featuressobel1	DYNGradient Featuressobel2	DYNGradient Featuressobel3
DYNGradient Featuressobel4	DYNGradient Featuressobel5	DYNGradient Featureskirsch
DYNGradient Featuresgrad1	DYNGradient Featuresgrad2	DYNnormalizedDyn
DYNsemi-quantitativeWI	DYNsemi-quantitativeWO	DYNsemi-quantitativeA
DYNsemi-quantitativeTa	DYNsemi-quantitativeSIonset	DYNsemi-quantitativeSImax
DYNsemi-quantitativeSIend	DYNsemi-quantitativeTonset	DYNsemi-quantitativeTmax
DYNsemi-quantitativeArea		

Table 8: List of the 20 features that were removed from the original set of features extracted from the 1.5T and 3T MRI.

Feature removed	**Reason**
T2Haralick featurescoorp	because same as T2Haralick featurescorrm
T2Haralick featuresdvarh	because same as T2Haralick featurescontr
ADCHaralick featurescontr	because too much value 0 for 3T
ADCHaralick featurescorrm	because too much value 1 for 3T
ADCHaralick featurescoorp	because too much value 1 for 3T
ADCHaralick featurescprom	because too much different between 1.5T and 3T
ADCHaralick featurescshad	because too much value 0 for 3T
ADCHaralick featuresdissi	because too much value 1 for 3T
ADCHaralick featuresenerg	because too much value 0 for 3T and 1.5T
ADCHaralick featureshomom	because too much value 1 for 3T
ADCHaralick featureshomop	because too much value 1 for 3T
ADCHaralick featuresmaxpr	because too much value 0 for 3T
ADCHaralick featuresdvarh	because too much value 0 for 3T
ADCHaralick featuresdenth	because too much value 1 for 3T
ADCHaralick featuresinf1h	because too much value 0 for 3T
ADCHaralick featuresinf2h	because too much value 1 for 3T
ADCHaralick featuresindnc	because too much value 1 for 3T
ADCHaralick featuresidmnc	because too much value 1 for 3T
DYNHaralick featurescoorp	because same as DYNHaralick featurescorrm
DYNHaralick featuresdvarh	because same as DYNHaralick featurescontr

B Results feature selection dataset Office Caltech

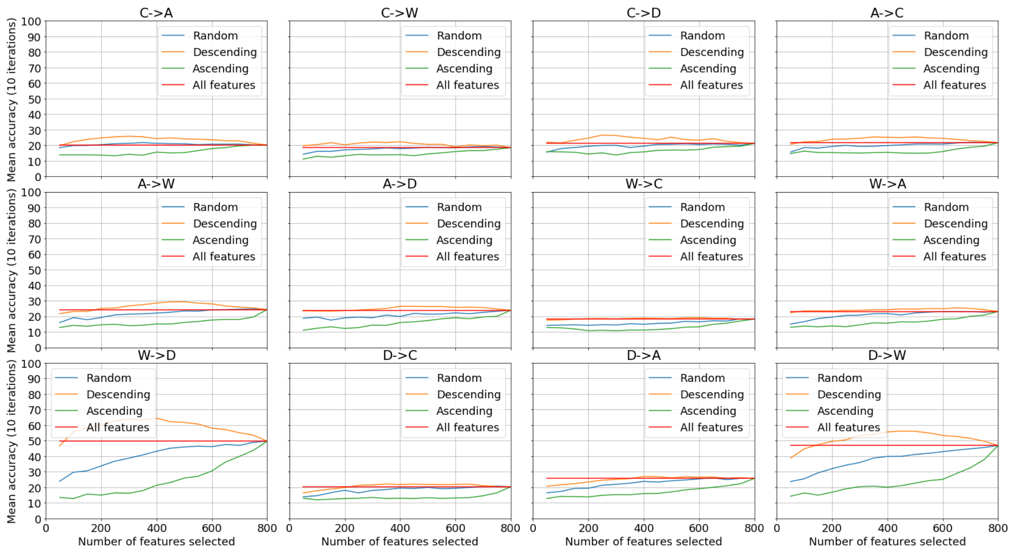


Figure 15: Adaptation algorithm: no adaptation. Image encoding: surf histograms.

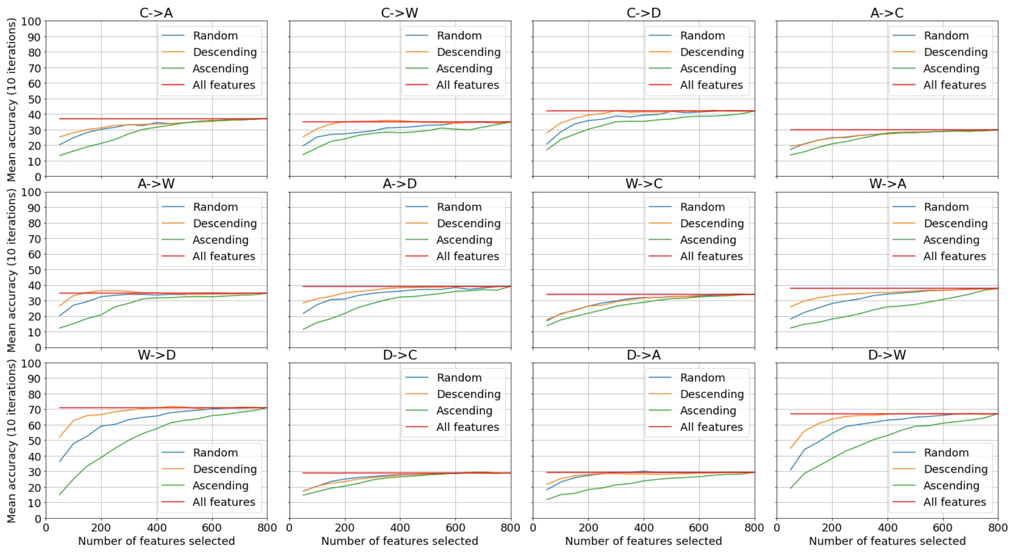


Figure 16: Adaptation algorithm: US_OT3 [4]. Image encoding: surf histograms.

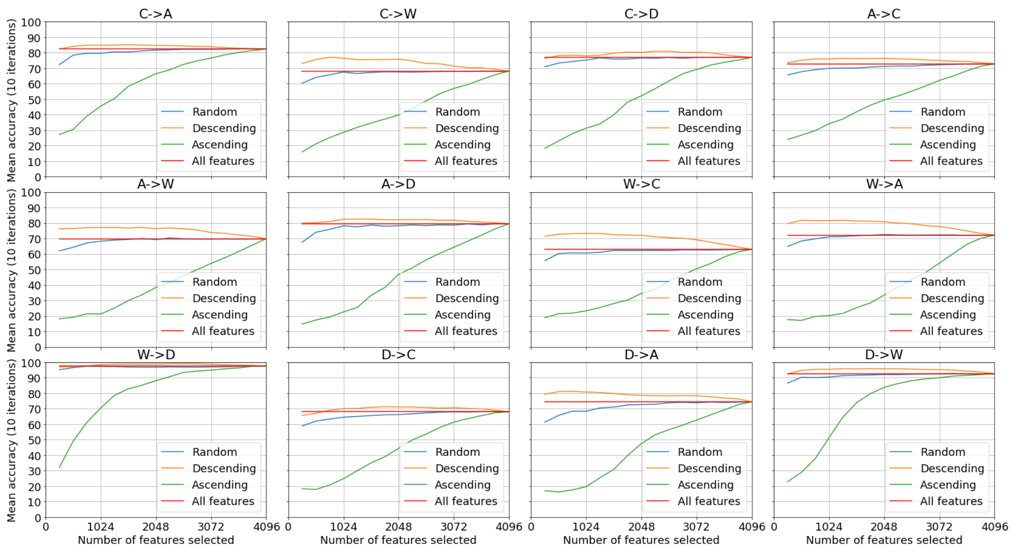


Figure 17: Adaptation algorithm: no adaptation. Image encoding: using pretrained CaffeNet [19].

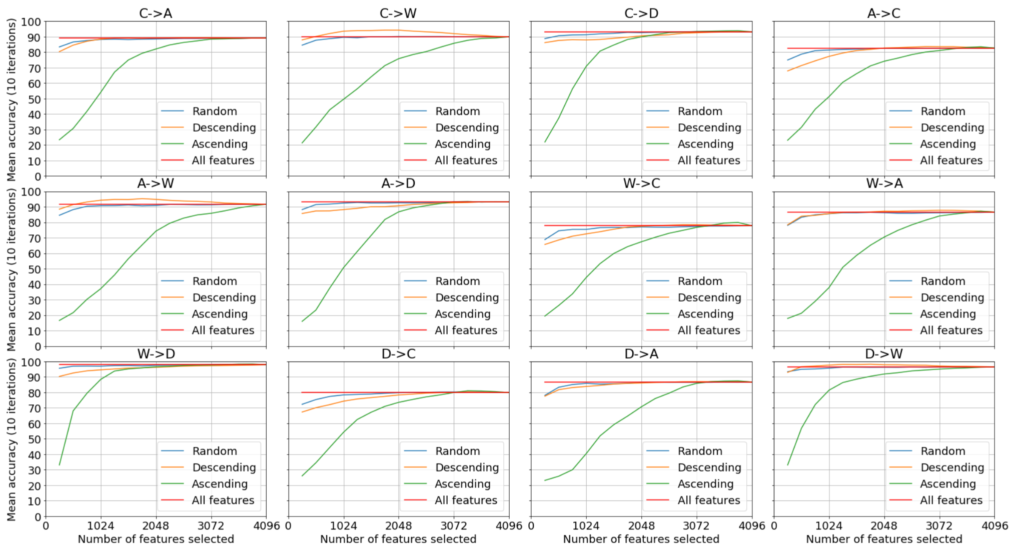


Figure 18: Adaptation algorithm: US_OT3 [4]. Image encoding: using pretrained CaffeNet [19].

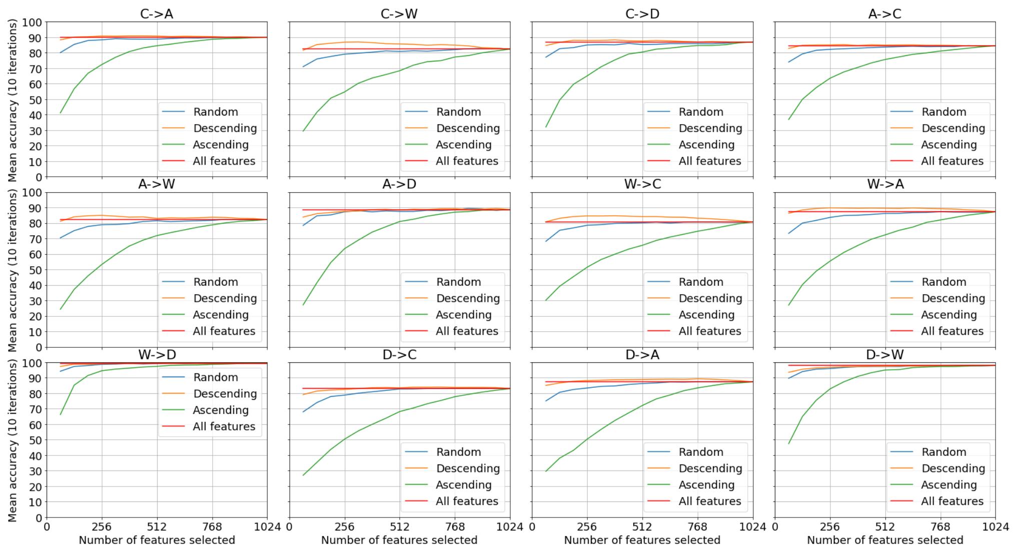


Figure 19: Adaptation algorithm: no adaptation. Image encoding: using pretrained GoogleNet [20].

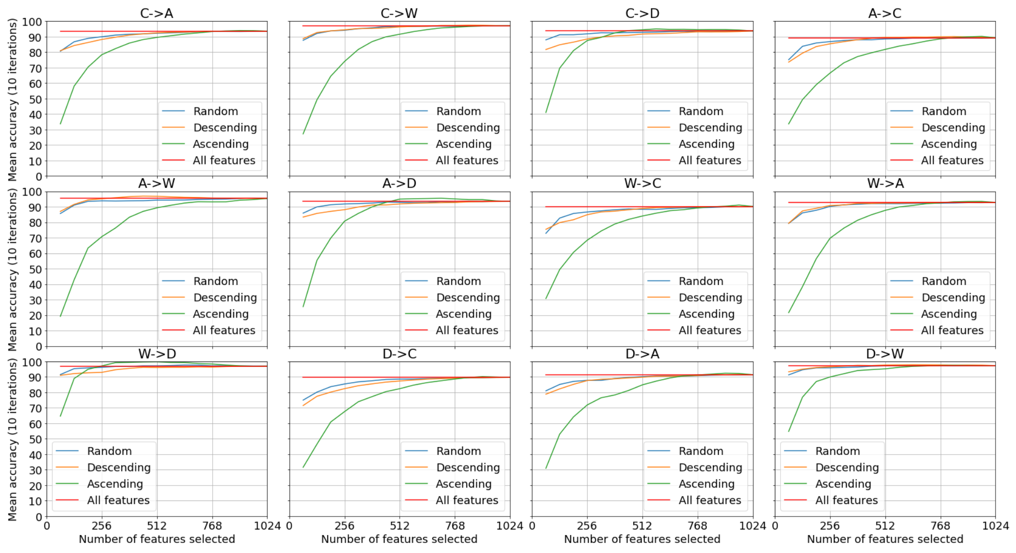


Figure 20: Adaptation algorithm: US_OT3 [4]. Image encoding: using pretrained GoogleNet [20].