Engineer's Thesis

Wojciech Pratkowiecki

December 2018

1 Visualization

To get some intuitions about how Gradient Reversal Layer and Conceptors work I tried to visualise the output of feature extractor. To do so I added a linear layer G_3 that takes the output vector $f \in R^D$ and maps it to $f' \in R^3$. The input vector x is firstly passed through feature extractor which results with $G_f(x)$ and then mapped into 3-dimensional space as $G_3(G_f(x))$. Now we can treat it as a 3-dimensional point.

1.1 MNIST and MNIST-M

Training model with such a small layer inside is much more difficult task, as many information may be lost during mapping do R^3. Therefore at the beginning I tried to get a simple MNIST classifier with no Gradient Reversal Layer. Fortunately the model achieved 98% accuracy on the test set, but it performed poorly on MNIST-M - it managed to recognize just 31% of digits, while with no 3D layer we can easily get a model with 61% accuracy. Now we can check how the input space X looks after mapping by feature extractor. Figure 1(a) shows the result of mapping for all samples from MNIST test set (blue) and MNIST-M test set (gray). Each point represents coordinates of vector obtained by passing it through feature extractor, so all of them can be described as $F_i = \{G_3(G_f(x)) | x \in X_{test_i}\}$, where $i \in \{\text{MNIST, MNIST-M}\}$. The shape of F_i resembles a multi-armed star. Moreover it seems like F_{MNIST} is just a stretched version of $F_{MNIST-M}$. We can expect that each arm contains mapped pictures of same digit. As we can see in Figure 1(b) zeros from both source and target distributions lie in the same direction and are represented by corresponding arm of stars. With this observation we can guess, that the domain adaptation problem in this case is a task of stretching the target space.

Our first approach could be using of conceptor computed with samples from F_{MNIST}. As we assume, the conceptor defines an ellipsoid that approximates points cloud from certain distribution. Therefore multiplying some set of points by the concpetor matrix C_{MNIST} would project the set into space of F_{MNIST}. Unfortunately in 3-dimensional case, the C_{MNIST} is almost an identity matrix I_3. We should not be surprised, as our data points cloud is a mulit-armed star

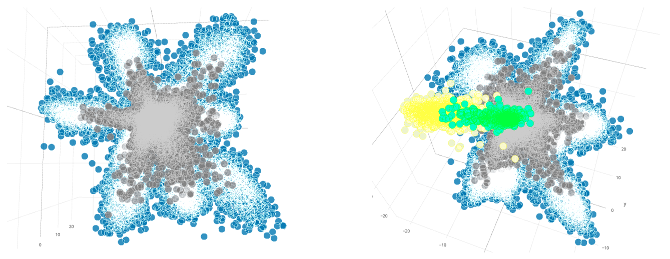
(a) MNIST and MNIST-M (b) Zeros of datasets

Figure 1: The visualization of samples transformed by feature extractor with output size equals 3. (a) presents two stars formed by samples from source domain (MNIST/blue) and a target domain (MNIST-M/gray) (b) shows how pictures of zeros are mapped - yellow points are zeros from MNIST and green ones from MNIST-M

and limiting it with an ellipsoid requires it to be a sphere. We cannot approximate those points with any kind of flatten ellipsoid. Moreover, the singular value decomposition (SVD) of conceptor matrix C results in $C = U\Sigma U^T$, where U and U^T can be interpreted as rotation matrices and Σ is a diagonal matrix with singular values of C on diagonal. As all singular values of conceptor matrix are real numbers from $[0, 1]$, the conceptor multiplying is in fact rotation, shortening (as non-zero values of Σ are ≤ 1) in certain directions and then re-rotating points (see Figure 2). Therefore we cannot transform $F_{MNIST-M}$ to resemble F_{MNIST} with conceptor matrix C_{MNIST}, as the points cloud would get even smaller, so it would increase the difference between source and target domains.

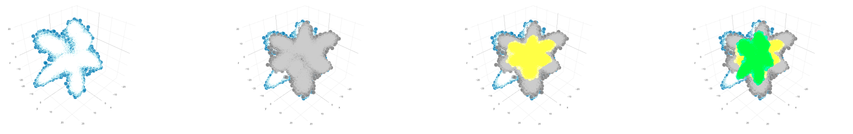
(a) F_M (b) $U^T \cdot F_M$ (c) $\Sigma \cdot U^T \cdot F_M$ (d) $U \cdot \Sigma \cdot U^T \cdot F_M$

Figure 2: The visualization of transformation points cloud by its conceptor based on SVD of concpetor matrix C. The representation F_{MNIST} of MNIST test set (blue points) has firstly been rotated (gray), then scaled (yellow) and finally re-rotated (green)

With this kind of knowledge one thing we can do is not transforming the $F_{MNIST-M}$ with C_{MNIST} but rather transforming points from F_{MNIST} with conceptor matrix $C_{MNIST-M}$ that describes an ellipse approximating points cloud $F_{MNIST-M}$. As mentioned before, the concpetor matrix is making points cloud tighter, so we could try get the bigger points cloud (the one corresponding to MNIST) and fit it to the size and shape of the smaller one (see Figure 3). As we already have a well-trained model that performs well in MNIST recognition, we can extend it with a layer with no learnable parameters, that gets the input and multiplies it by conceptor matrix $C_{MNIST-M}$. Formally our model so far is a sequence of feature extractor (with parameters Θ_f, 3D mapping and its parameters Θ_3 and label classifier G_y with parameters Θ_y, so the result for input vector x is $G_y(G_3(G_f(x;\Theta_f);\Theta_3);\Theta_y)$. Now we want to freeze the feature extractor and 3D mapping part of the architecture (let's call it G_3) and multiply its result by $C_{MNIST-M}$, what gives us the new representation ($G_3C = C_{MNIST-M} \cdot G_3$) of the input vector from source domain. We expect it to lie in the blue points cloud shown in the second picture of Figure 3. If the model constructed with non learnable G_3C and new-learned G_y learns to classify the digits from source domain we would expect the model containing G_3 and G_y perform well on target domain, as the G_3 transforms MNIST-M to grey points cloud from mentioned picture, so G_y should hardly see the difference between domains. After just few epochs the label classifier adjusted to new representation and managed to reach 98% accuracy on MNIST test set. Now we "unpack" the G_3C to G_3 and check its performance on MNIST-M. The model got 36% of accuracy, so it improved by 16% from previous 31%. We can see the difference, but we could expect something better, as we forced the domain shift to be insignificant. The lat picture of Figure 3 presents three points clouds - the blue one is the result of transforming source domain test set by G_3C. The yellow is target domain test set after transformation by G_3 (same as grey points from second picture). The green points cloud is a mapping target domain by G_3C. As conceptor matrix is making points cloud tighter this distribution was expected. It is worth mentioning that we can easily get a domain predictor G_{d_3} that is performing well on both G_3 and G_3C transformation - the accuracy of prediction whether the point belongs to target or source domain (equivalent to determinate if the point with given coordinates is blue or gray if we are considering distribution from first picture of Figure 3) reached 95%. If we look closer at the plot of points we could see that many of target domain ones lie closely to the middle of coordinate system, so maybe the mapping works fine for few examples which forms the multi-armed star, but most of samples are left with poor transformation, which is unclear for label classifier. Therefore after this test we can have few conclusions. Firstly we cannot use our conceptors intuitions from 3-dimensional space in d-dimensional one, as in 3D we are not able to approximate the multi-armed star obtained by samples with nothing but sphere, but we do not know how it behaves in higher dimensions. Nevertheless the improvement of predicting labels for target domain after including conceptor layer shows, that we should consider using conceptors during domain adaptation. The density of target domain points near the $(0, 0, 0)$ shows that those points clouds look similar at the first glance, but we cannot get a good classifier for target domain with this kind of representation, because we can expect that pictures of different digits lie really closely there. With those knowledge next things we can do to have a better understanding of our problem is trying to get a model with Gradient Reversal Layer in 3-dimensional space, and check the performance of conceptor layer in higher dimensions. Moreover we can check the variant with mapping feature extractor to 2-dimensions to verify all noticed behaviours.

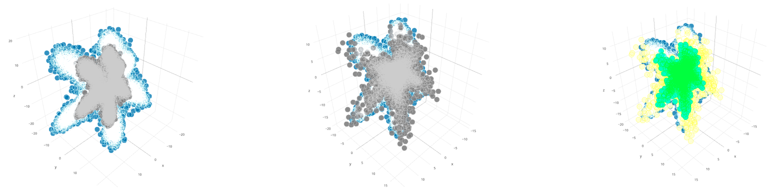

Figure 3: Transforming data with conceptor matrix $C_{MNIST-M}$ corresponding to target domain samples. First picture shows the 3D mapping of source domain (blue) and its transformation. The second one presents the transformation G_3C of samples form source domain and 3D mapping G_3 of target domain test set. Last picture shows distributions after training label classifier on G_3C mapping - blue points are again transformation of source domain, yellow ones are 3D mapping of target domain (equivalent to grey points from previous picture) and green points cloud is its G_3C transformation.