# EFFICIENT DOMAIN GENERALIZATION VIA COMMON-SPECIFIC LOW-RANK DECOMPOSITION

**Vihari Piratla**
MSR India

**Praneeth Netrapalli**
MSR India

**Sunita Sarawagi**
IIT Bombay

## ABSTRACT

Domain generalization refers to the task of training a model which generalizes to new domains that are not seen during training. We present CSD (Common Specific Decomposition), for this setting, which *jointly* learns a common component (which generalizes to new domains) and a domain specific component (which overfits on training domains). The domain specific components are discarded after training and only the common component is retained. The algorithm is extremely simple and involves only modifying the final linear classification layer of any given neural network architecture. We present a principled analysis to understand existing approaches, provide identifiability results of CSD, and study effect of low-rank on domain generalization. We show that CSD either matches or beats state of the art approaches for domain generalization based on domain erasure, domain perturbed data augmentation, and meta-learning. Further diagnostics on rotated MNIST, where domains are interpretable, confirm the hypothesis that CSD successfully disentangles common and domain specific components and hence leads to better domain generalization. The code and datasets can be found at the following URL: `https://github.com/vihari/CSD`.

## 1 Introduction

In the domain generalization (DG) task we are given domain-demarcated data from multiple domains during training, and our goal is to create a model that will generalize to instances from new domains during testing. Unlike in the more popular domain adaptation task Mansour et al. (2009); Ben-David et al. (2006); DauméIII et al. (2010) that explicitly adapts to a fixed target domain, DG requires zero-shot generalization to individual instances from multiple new domains. Domain generalization is of particular significance in large-scale deep learning networks because large training sets often entail aggregation from multiple distinct domains, on which today's high-capacity networks easily overfit. Standard methods of regularization only address generalization to unseen instances sampled from the distribution of training domains, and have been shown to perform poorly on instances from unseen domains.

Domain Generalization approaches have a rich history. Earlier methods were simpler and either sought to learn feature representations that were invariant across domains Motiian et al. (2017); Muandet et al. (2013); Ghifary et al. (2015); Wang et al. (2019), or decomposed parameters into shared and domain-specific components Khosla et al. (2012); Li et al. (2017). Of late however the methods proposed for DG are significantly more complicated and expensive. A recent favorite is gradient-based meta-learning that trains with sampled domain pairs to minimize either loss on one domain while updating parameters on another domain Balaji et al. (2018); Li et al. (2018a), or minimizes the divergence between their representations Dou et al. (2019). Another approach is to augment training data with domain adversarial perturbations Shankar et al. (2018).

**Contributions**
Our paper starts with analyzing the domain generalization problem in a simple and natural generative setting. We use this setting to provide a principled understanding of existing DG approaches and improve upon prior decomposition methods. We design an algorithm CSD that decomposes only the last softmax parameters into a common component and a low-rank domain-specific component with a regularizer to promote orthogonality of the two parts. We prove identifiability of the shared parameters, which was missing in earlier decomposition-based approaches. We analytically

study the effect of rank in trading off domain-specific noise suppression and domain generalization, which in earlier work was largely heuristics-driven.

We show that our method is almost an order of magnitude faster than state-of-the-art meta-learning based methods Dou et al. (2019), and provides higher accuracy than existing approaches, particularly when the number of domains is large. Our experiments span both image and speech datasets and a large range of training domains (5 to 1000), in contrast to recent DG approaches evaluated only on a few domains. We provide empirical insights on the working of CSD on the rotated MNIST datasets where the domains are interpretable, and show that CSD indeed manages to separate out the generalizable shared parameters while training with simple domain-specific losses. We present an ablation study to evaluate the importance of the different terms in our training objective that led to improvements with regard to earlier decomposition approaches.

## 2 Related Work

The work on Domain Generalization is broadly characterized by four major themes:

**Domain Erasure** Many early approaches attempted to repair the feature representations so as to reduce divergence between representations of different training domains. Muandet et al. (2013) learns a kernel-based domain-invariant representation. Ghifary et al. (2015) estimates shared features by jointly learning multiple data-reconstruction tasks. Li et al. (2018b) uses MMD to maximize the match in the feature distribution of two different domains. The idea of domain erasure is further specialized in Wang et al. (2019) by trying to project superficial (say textural) features out using image specific kernels. Domain erasure is also the founding idea behind many domain adaptation approaches, example Ben-David et al. (2006); Hoffman et al. (2018); Ganin et al. (2016) to name a few.

**Augmentation** The idea behind these approaches is to train the classifier with instances obtained by domains hallucinated from the training domains, and thus make the network 'ready' for these neighboring domains. Shankar et al. (2018) proposes to augment training data with instances perturbed along directions of domain change. A second classifier is trained in parallel to capture directions of domain change. Volpi et al. (2018) applies such augmentation on single domain data. Another type of augmentation is to simultaneously solve for an auxiliary task. For example, Carlucci et al. (2019) achieves domain generalization for images by solving an auxiliary unsupervised jig-saw puzzle on the side.

**Meta-Learning/Meta-Training** A recent popular approach is to pose the problem as a meta-learning task, whereby we update parameters using meta-train loss but simultaneously minimizing meta-test loss Li et al. (2018a),Balaji et al. (2018) or learn discriminative features that will allow for semantic coherence across meta-train and meta-test domains Dou et al. (2019). More recently, this problem is being pursued in the spirit of estimating an invariant optimizer across different domains and solved by a form of meta-learning in Arjovsky et al. (2019). Meta-learning approaches are complicated to implement, and slow to train.

**Decomposition** In these approaches the parameters of the network are expressed as the sum of a common parameter and domain-specific parameters during training. DauméIII (2007) first applied this idea for domain adaptation. Khosla et al. (2012) applied decomposition to DG by retaining only the common parameter for inference. Li et al. (2017) extended this work to CNNs where each layer of the network was decomposed into common and specific low-rank components. Our work provides a principled understanding of when and why these methods might work and uses this understanding to design an improved algorithm CSD. Three key differences are: CSD decomposes only the last layer, imposes loss on both the common and domain-specific parameters, and constrains the two parts to be orthogonal. We show that orthogonality is required for theoretically proving identifiability. As a result, this newer avatar of an old decomposition-based approach surpasses recent, more involved augmentation and meta-learning approaches.

## 3 Our Approach

Our approach is guided by the following assumption about domain generalization settings.

**Assumption**: There are common features in the data whose correlation with label is consistent across domains and domain specific features whose correlation with label varies (from positive to negative) across domains. Classifiers that rely on common features generalize to new unseen domains far better than those that rely on domain specific features.

Note that we make *no* assumptions about a) the domain predictive power of common features and b) the net correlation between domain specific features and the label. Let us consider the following simple example which illustrates these

points. There are $D$ training domains and examples $(x, y)$ from domain $i \in [D]$ are generated as follows:

$$x = y(e_c + \beta_i e_s) + \mathcal{N}(0, \Sigma_i) \in \mathbb{R}^m, \ \forall \ i \in [D] \tag{1}$$

where $y = \pm 1$ with equal probability, $m$ is the dimension of the training examples, $e_c \in \mathbb{R}^m$ is a common feature whose correlation with the label is constant across domains and $e_s \perp e_c \in \mathbb{R}^m$ is a domain specific feature whose correlation with the label, given by the coefficients $\beta_i$, varies from domain to domain. In particular, for each domain $i$, suppose $\beta_i \sim \text{Unif}[-1, 2]$. Note that though $\beta_i$ vary from positive to negative across various domains, there is a net positive correlation between $e_s$ and the label $y$. $\mathcal{N}(0, \Sigma_i)$ denotes a standard normal random variable with mean zero and covariance matrix $\Sigma_i$. Since $\Sigma_i$ varies across domains, every feature captures some domain information. We note that the assumption $e_s \perp e_c$ is indeed restrictive – we use it here only to make the discussion and expressions simpler. We relax this assumption later in this section when we discuss the identifiability of domain generalizing classifier.

Our assumption at the beginning of this section (which envisages the possibility of seeing $\beta_i \notin [-1, 2]$ at test time) implies that the only classifier that generalizes to new domains is one that depends solely on $e_c$ [1]. Consider training a linear classifier on this dataset. We will describe the issues faced by existing domain generalization methods.

- **Empirical risk minimization (ERM)**: When we empirically train a linear classifier using ERM with cross entropy loss on all of the training data, the resulting classifier puts significant nonzero weight on the domain specific component $e_s$. The reason for this is that there is a bias in the training data which gives an overall positive correlation between $e_s$ and the label.

- **Domain erasure** Ganin et al. (2016): Domain erasure methods seek to extract features that have the same distribution across different domains and construct a classifier using those features. The difference in noise variance in (1) across domains means that all features have domain signal. In fact, linear classifiers on any feature can obtain nontrivial domain classification accuracy. The premise of domain erasure methods, that there exist features which have high prediction power of the label but do not capture domain information, does not apply in this setting and domain erasure methods do not perform well.

- **Domain adversarial perturbations Shankar et al. (2018)**: Domain adversarial perturbations seek to augment the training dataset with adversarial examples obtained using domain classification loss, and train a classifier on the resulting augmented dataset. Since the common component $e_c$ has domain signal, the adversarial examples will induce variation in this component and so the resulting classifier puts less weight on the common component.

- **Meta-learning**: Meta-learning based DG approaches such as Dou et al. (2019) work with pairs of domains. Parameters updated using gradients on loss of one domain, when applied on samples of both domains in the pair should lead to similar class distributions. If the method used to detect similarity is robust to domain-specific noise, meta-learning methods could work well in this setting. But meta-learning methods require second order gradient updates, and/or are generally considered expensive to implement.

**Decomposition based approaches Khosla et al. (2012); Li et al. (2017)**: Decomposition based approaches rely on the observation that for problems like (1), there exist good domain specific classifiers $w_i$, one for each domain $i$, such that:

$$\tilde{w}_i = e_c + \gamma_i e_s, \tag{2}$$

where $\gamma_i$ is a function of $\beta_i$. Note that all these domain specific classifiers share the common component $e_c$ which is the domain generalizing classifier that we are looking for! If we are able to find domain specific classifiers of the form (2), we can extract $e_c$ from them. This idea can be extended to a generalized version of (1), where the latent dimension of the domain space is $k$ i.e., say

$$x = y(e_c + \sum_{j=1}^{k} \beta_{i,j} e_{s_j}) + \mathcal{N}(0, \Sigma_i). \tag{3}$$

$e_{s_j} \perp e_c \in \mathbb{R}^m$, and $e_{s_j} \perp e_{s_\ell}$ for $j, \ell \in [k], j \neq \ell$ are domain specific features whose correlation with the label, given by the coefficients $\beta_{i,j}$, varies from domain to domain. In this setting, there exist good domain specific classifiers $\tilde{w}_i$ such that:

$$\tilde{w}_i = e_c + E_s \gamma_i$$

---

[1]Note that this last statement relies on the assumption that $e_c \perp e_s$. If this is not the case, the correct domain generalizing classifier is the component of $e_c$ that is orthogonal to $e_s$ i.e., $e_c - \frac{\langle e_c, e_s \rangle}{\|e_s\|^2} \cdot e_s$. See (6).

where $e_c \in \mathbb{R}^m$ is a domain generalizing classifier, $E_s = \begin{bmatrix} e_{s_1} & e_{s_2} & \cdots & e_{s_k} \end{bmatrix} \in \mathbb{R}^{m \times k}$ consists of domain specific components and $\gamma_i \in \mathbb{R}^k$ is a domain specific combination of the domain specific components that depends on $\beta_{i,j}$ for $j = 1, \cdots, k$. With this observation, the algorithm is simple to state: train domain specific classifiers $\tilde{w}_i$ that can be represented as

$$\tilde{w}_i = w_c + W_s \gamma_i \in \mathbb{R}^m. \tag{4}$$

Here the training variables are $w_c \in \mathbb{R}^m, W_s \in \mathbb{R}^{m \times k}$ and $\gamma_i \in \mathbb{R}^k$. After training, discard all the domain specific components $W_s$ and $\gamma_i$ and return the common classifier $w_c$. Note that (4) can equivalently be written as

$$W = w_c \mathbb{1}^\top + W_s \Gamma^\top, \tag{5}$$

where $W := \begin{bmatrix} \tilde{w}_1 & \tilde{w}_2 & \cdots & \tilde{w}_D \end{bmatrix}$, $\mathbb{1} \in \mathbb{R}^D$ is the all ones vector and $\Gamma^\top := \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_D \end{bmatrix}$.

This framing of the decomposition approach, in the context of simple and concrete examples as in (1) and (3), lets us understand the three main aspects that are not properly addressed by prior works in this space: 1) identifiability of $w_c$, 2) choice of low rank and 3) extension to non-linear models such as neural networks.

**Identifiability of the common component** $w_c$: None of the prior decomposition based approaches investigate identifiability of $w_c$. In fact, given a general matrix $W$ which can be written as $w_c \mathbb{1}^\top + W_s \Gamma^\top$, there are multiple ways of decomposing $W$ into this form, so $w_c$ cannot be uniquely determined by this decomposition alone. For example, given a decomposition (5), for any $(k+1) \times (k+1)$ invertible matrix $R$, we can write $W = \begin{bmatrix} w_c & W_s \end{bmatrix} R^{-1} R \begin{bmatrix} \mathbb{1} & \Gamma \end{bmatrix}^\top$. As long as the first row of $R$ is equal to $\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$, the structure of the decomposition (5) is preserved while $w_c$ might no longer be the same. Out of all the different $w_c$ that can be obtained this way, which one is the *correct domain generalizing classifier*? In the setting of (3), where $e_c \perp E_s$, we proposed that the correct domain generalizing classifier is $w_c = e_c$. In the setting where $e_c \not\perp E_s$, we propose that the correct domain generalizing classifier is the projection of $e_c$ onto the space orthogonal to Span $(E_s)$ i.e.,

$$w_c = e_c - P_{E_s} e_c, \tag{6}$$

where $P_{E_s}$ is the projection matrix onto the span of the domain specific vectors $e_s$. The following lemma characterizes this condition in terms of the decomposition (5).

**Lemma 1.** *Suppose $W := e_c \mathbb{1}^\top + E_s \hat{\Gamma}^\top = w_c \mathbb{1}^\top + W_s \Gamma^\top$ is a rank-$(k+1)$ matrix, where $E_s \in \mathbb{R}^{m \times k}, \hat{\Gamma} \in \mathbb{R}^{D \times k}, W_s \in \mathbb{R}^{m \times k}$ and $\Gamma \in \mathbb{R}^{D \times k}$ are all rank-$k$ matrices with $k < m, D$. Then, $w_c = e_c - P_{E_s} e_c$ if and only if $w_c \perp$ Span $(W_s)$.*

*Proof.* **If direction**: Suppose $w_c \perp$ Span $(W_s)$. Then, $W^\top w_c = \langle e_c, w_c \rangle \cdot \mathbb{1} + \hat{\Gamma} \cdot \left( E_s^\top w_c \right) = \|w_c\|^2 \cdot \mathbb{1}$. Since $W$ is a rank-$(k+1)$ matrix, we know that $\mathbb{1} \notin$ Span $\left( \hat{\Gamma} \right)$ and so it has to be the case that $\langle e_c, w_c \rangle = \|w_c\|^2$ and $E_s^\top w_c = 0$. Both of these together imply that $w_c$ is the projection of $e_c$ onto the space orthogonal to $E_s$ i.e., $w_c = e_c - P_{E_s} e_c$.

**Only if direction**: Let $w_c = e_c - P_{E_s} e_c$. Then $e_c \mathbb{1}^\top - w_c \mathbb{1}^\top + E_s \hat{\Gamma}^\top = P_{E_s} e_c \mathbb{1}^\top + E_s \hat{\Gamma}^\top$ is a rank-$k$ matrix and can be written as $W_s \Gamma^\top$ with Span $(W_s) =$ Span $(E_s)$. Since $w_c \perp$ Span $(E_s)$, we also have $w_c \perp$ Span $(W_s)$. $\qquad \square$

So we train for classifiers (5) satisfying $w_c \perp$ Span $(W_s)$.

**Why low rank?**: An important choice in the decomposition approaches is the *low* rank of the decomposition (5), which in prior works was justified heuristically, by appealing to number of parameters. We prove the following result, which gives us a more principled reason for the choice of low rank parameter $k$.

**Theorem 1.** *Given any matrix $W \in \mathbb{R}^{m \times D}$, the minimizers of the function $f(w_c, W_s, \Gamma) = \left\| W - w_c \mathbb{1}^\top - W_s \Gamma^\top \right\|_F^2$, where $W_s \in \mathbb{R}^{m \times k}$ and $w_c \perp$ Span $(W_s)$ can be computed by the following steps:*

- $w_c \leftarrow \frac{1}{D} W \cdot \mathbb{1}$.

- $W_s, \Gamma \leftarrow$ *Top-k SVD* $\left( W - w_c \mathbb{1}^\top \right)$.

- $w_c^{new} \leftarrow \frac{1}{\left\| (w_c \mathbb{1}^\top + W_s \Gamma^\top)^+ \mathbb{1} \right\|^2} \left( w_c \mathbb{1}^\top + W_s \Gamma^\top \right)^+ \mathbb{1}$.

- $W_s^{new} \Gamma^{new \top} \leftarrow w_c \mathbb{1}^\top + W_s \Gamma^\top - w_c^{new} \mathbb{1}^\top$.

- *Output* $w_c^{new}, W_s^{new}, \Gamma^{new}$

4

The proof of this theorem is similar to that of the classical low rank approximation theorem of Eckart-Young-Mirsky, and is presented in the supplementary material. As special cases of the above result, we see that for $k = 0$, we just obtain the average classifier over all domains $w_c = \frac{1}{D} W \cdot \mathbb{1}$, while for $k = D - 1$, we obtain $w_c = W^+ \mathbb{1} / \|W^+ \mathbb{1}\|^2$. When $W = w_c \mathbb{1}^\top + W_s \Gamma + N$, where $N$ is a noise matrix (for example due to finite samples), both extremes $k = 0$ and $k = D - 1$ have different advantages/drawbacks:

- $k = 0$: The averaging effectively reduces the noise component $N$ but ends up retaining some domain specific components if there is net correlation with the label in the training data.

- $k = D - 1$: The pseudoinverse effectively removes domain specific components and retains only the common component (by Theorem 1). However, the pseudoinverse does not reduce noise to the same extent as a plain averaging would (since empirical mean is often asymptotically the best estimator for mean).

In general, the sweet spot for $k$ lies between 0 and $D - 1$ and its precise value depends on the dimension and magnitude of the domain specific components as well as the magnitude of noise. In our implementation, we perform cross validation to choose a good value for $k$ but also note that the performance of our algorithm is relatively stable with respect to this choice in Section 4.3.

**Extension to neural networks**: Finally, prior works extend this approach to non-linear models such as neural networks by imposing decomposition of the form (5) for parameters in all layers separately. This increases the size of the model significantly and leads to worse generalization performance. Further, it is not clear whether any of the insights we gained above for linear models continue to hold when we include non-linearities and stack them together. So, we propose the following two simple modifications instead:

- enforcing the structure (5) only in the final linear layer, as opposed to the standard single softmax layer, and

- including a loss term for predictions of common component, in addition to the domain specific losses,

both of which encourage learning of *features* with common-specific structure.

Our experiments (Section 4.2) show that these modifications (orthogonality, changing only the final linear layer and including common loss) are instrumental in making decomposition methods state of the art for domain generalization. Our overall training algorithm below details the steps.

## 3.1 Algorithm CSD

Our method of training neural networks for domain generalization appears as Algorithm 1 and is called CSD for Common-specific Decomposition. The analysis above was for the binary setting, but we present the algorithm for the multi-class case with $C = \#$ classes. The only extra parameters that CSD requires, beyond normal feature parameters $\theta$ and softmax parameters $w_c \in \mathbb{R}^{C \times m}$, are the domain-specific low-rank parameters $W_s \in \mathbb{R}^{C \times m \times k}$ and $\gamma_i \in \mathbb{R}^k$, for $i \in [D]$. Here $m$ is the representation size in the penultimate layer. Thus, $\Gamma^\top = [\gamma_1, \ldots, \gamma_D]$ can be viewed as a domain-specific embedding matrix of size $k \times D$. Note that unlike a standard mixture of softmax, the $\gamma_i$ values are not required to be on the simplex. Each training instance consists of an input $x$, true label $y$, and a domain identifier $i$ from 1 to $D$. Its domain-specific softmax parameter is computed by $w_i = w_c + W_s \gamma_i$.

Instead of first computing the full-rank parameters and then performing SVD, we directly compute the low-rank decomposition along with training the network parameters $\theta$. For this we add a weighted combination of these three terms in our training objective:

(1) Orthonormality regularizers to make $w_c[y]$ orthogonal to domain-specific $W_s[y]$ softmax parameters for each label $y$ and to avoid degeneracy by controlling the norm of each softmax parameter to be close to 1.

(2) A cross-entropy loss between $y$ and distribution computed from the $w_i$ parameters to train both the common and low-rank domain-specific parameters.

(3) A cross-entropy loss between $y$ and distribution computed from the $w_c$ parameters. This loss might appear like normal ERM loss but when coupled with the orthogonality regularizer above it achieves domain generalization.

## 3.2 Synthetic setting: comparing CSD with ERM

We use the data model proposed in Equation 1 to simulate multi-domain training data with $D = 10$ domains and $m = 2$ features. For each domain, we sample $\beta_i$ uniformly from -1, 2 and $\sigma_{ij}$ uniformly from 0, 1. We set $e_c = [1, 0]$ and $e_s = [0, 1]$. We sample 100 data points for each domain using its corresponding values: $\beta_i, \Sigma_i$. We then fit the

---

**Algorithm 1** Common-Specific Low-Rank Decomposition (CSD )

---

1: **Given:** $D, m, k, C, \lambda, \kappa$,train-data
2: Initialize params $w_c \in \mathbb{R}^{C \times m}, W_s \in \mathbb{R}^{C \times m \times k}$
3: Initialize $\gamma_i \in \mathbb{R}^k : i \in [D]$
4: Initialize params $\theta$ of feature network $G_\theta : \mathcal{X} \mapsto \mathbb{R}^m$
5: $\hat{W} = [w_c^T, W_s^T]^T$
6: $\mathcal{R} \leftarrow \sum_{y=1}^{C} \|I_{k+1} - \hat{W}[y]^T \hat{W}[y]\|_F^2$            ▷ Orthonormality constraint
7: **for** $(x, y, i) \in$ train-data **do**
8:      $w_i \leftarrow w_c + W_s \gamma_i$
9:      loss $+= \mathcal{L}(G_\theta(x), y; w_i) + \lambda \mathcal{L}(G_\theta(x), y; w_c)$
10: **end for**
11: Optimize loss$+\kappa\mathcal{R}$ wrt $\theta, w_c, W_s, \gamma_i$
12: **Return** $\theta, w_c$                                                  ▷ **for inference**

---

parameters of a linear classifier with log loss using either standard expected risk minimization (ERM) estimator or CSD .

The scaled solution obtained using ERM is [1, 0.2] and [1, 0.03] using CSD with high probability from ten runs. As expected, the solution of ERM has a positive but small coefficient on the second component due to the net positive correlation on this component. CSD on the other hand correctly decomposed the common component.

# 4 Experiments

We compare our method with three existing domain generalization methods: (1) **MASF** Dou et al. (2019) is a recently proposed meta-learning based strategy to learn domain-invariant features. (2) **CG**: As a representative of methods that augment data for domain generalization we compare with Shankar et al. (2018), and (3) **LRD**: the low-rank decomposition approach of Li et al. (2017) but only at the last softmax layer. Our baseline is standard expected risk minimization (ERM) using cross-entropy loss that ignores domain boundaries altogether.

We evaluate on five different datasets spanning image and speech data types and varying number of training domains. We assess quality of domain generalization as accuracy on a set of test domains that are disjoint from the set of training domains.

**Experiment setup details** We use ResNet-18 to evaluate on rotated image tasks, LeNet for Handwritten Character datasets, and a multi-layer convolution network similar to what was used for Speech tasks in Shankar et al. (2018). We added a layer normalization just before the final layer in all these networks since it helped generalization error on all methods, including the baseline. CSD is relatively stable to hyper-parameter choice, we set the default rank to 1, and parameters of weighted loss to $\lambda = 1$ and $\kappa = 1$. These hyper-parameters along with learning rates of all other methods as well as number of meta-train/meta-test domains for MASF and step size of perturbation in CG are all picked using a task-specific development set. Further, we scale $\Gamma$ using sigmoid activation.

**Handwritten character datasets:** In these datasets we have characters written by many different people, where the person writing serves as domain and generalizing to new writers is a natural requirement. Handwriting datasets are challenging since it is difficult to disentangle a person's writing style from the character (label), and methods that attempt to erase domains are unlikely to succeed. We have two such datasets.

(1) The LipitK dataset[2] earlier used in Shankar et al. (2018) is a Devanagari Character dataset which has classification over 111 characters (label) collected from 106 people (domain). We train three different models on each of 25, 50, and 76 domains, and test on a disjoint set of 20 domains while using 10 domains for validation.

(2) Nepali Hand Written Character Dataset (NepaliC)[3] contains data collected from 41 different people on consonants as the character set which has 36 classes. Since the number of available domains is small, in this case we create a fixed split of 27 domains for training, 5 for validation and remaining 9 for testing.

We use LeNet as the base classifier on both the datasets.

In Table 1 we show the accuracy using different methods for different number of training domains on the LipitK dataset, and on the Nepali dataset. We observe that across all four models CSD provides significant gains in accuracy over the

---

[2] http://lipitk.sourceforge.net/datasets/dvngchardata.htm
[3] https://www.kaggle.com/ashokpant/devanagari-character-dataset

|  | LipitK | | | NepaliC |
| --- | --- | --- | --- | --- |
| Method | 25 | 50 | 76 | 27 |
| ERM (Baseline) | 74.5 (0.4) | 83.2 (0.8) | 85.5 (0.7) | 83.4 (0.4) |
| LRD Li et al. (2017) | 76.2 (0.7) | 83.2 (0.4) | 84.4 (0.2) | 82.5 (0.5) |
| CG Shankar et al. (2018) | 75.3 (0.5) | 83.8 (0.3) | 85.5 (0.3) | 82.6 (0.5) |
| MASF Dou et al. (2019) | **78.5** (0.5) | 84.3 (0.3) | 85.9 (0.3) | 83.3 (1.6) |
| CSD (Ours) | 77.6 (0.4) | **85.1** (0.6) | **87.3** (0.4) | **84.1** (0.5) |

Table 1: Comparison of our method on two handwritting datasets: LipitK and NepaliC. For LipitK since number of available training domains is large we also report results with increasing number of domains. The numbers are average (and standard deviation) from three runs.

baseline (ERM), and all three existing methods LRD, CG and MASF. The gap between prior decomposition-based approach (LRD) and ours, establishes the importance of our orthogonality regularizer and common loss term. MASF is better than CSD only for 25 domains and as the number of domains increases to 76, CSD's accuracy is 87.3 whereas MASF's is 85.9.

In terms of training time MASF is 5–10 times slower than CSD, and CG is 3–4 times slower than CSD. In contrast CSD is just 1.1 times slower than ERM. Thus, the increased generalization of CSD incurs little additional overheads in terms of training time compared to existing methods.

| Method | 50 | 100 | 200 | 1000 |
| --- | --- | --- | --- | --- |
| ERM | 72.6 (.1) | 80.0 (.1) | 86.8 (.3) | 90.8 (.2) |
| CG | 73.3 (.1) | 80.4 (.0) | 86.9 (.4) | 91.2 (.2) |
| CSD | **73.7** (.1) | **81.4** (.4) | **87.5** (.1) | **91.3** (.2) |

Table 2: Accuracy comparison on speech utterance data with varying number of training domains. The numbers are average (and standard deviation) from three runs.

**Speech utterances dataset** We use the utterance data released by Google which was also used in Shankar et al. (2018) and is collected from a large number of subjects[4]. The base classifier and the preprocessing pipeline for the utterances are borrowed from the implementation provided in the Tensorflow examples[5]. We used the default ten (of the 30 total) classes for classification similar to Shankar et al. (2018). We use ten percent of total number of domains for each of validation and test.

The accuracy comparison for each of the methods on varying number of training domains is shown in Table 2. We could not compare with MASF since their implementation is only made available for image tasks. Also, we skip comparison with LRD since earlier experiments established that it can be worse than even the baseline. Table 2 shows that CSD is better than both the baseline and CG on all domain settings. When the number of domains is very large (for example, 1000 in the table), even standard training can suffice since the training domains could 'cover' the variations in the test domains.

|  | MNIST | | Fashion-MNIST | |
| --- | --- | --- | --- | --- |
|  | in-domain | out-domain | in-domain | out-domain |
| ERM | 98.3 (0.0) | 93.6 (0.7) | 89.5 (0.1) | 76.5 (0.7) |
| MASF | 98.2 (0.1) | 93.2 (0.2) | 86.9 (0.3) | 72.4 (2.9) |
| CSD | **98.4** (0.0) | **94.7** (0.2) | **89.7** (0.2) | **78.9** (0.7) |

Table 3: Performance comparison on rotated MNIST and rotated Fashion-MNIST, shown are the in-domain and out-domain accuracies averaged over three runs along with standard deviation in the brackets.

|  | MNIST | |
| --- | --- | --- |
|  | in-domain | out-domain |
| ERM | 97.7 (0.) | 89.0 (.8) |
| MASF | 97.8 (0.) | 89.5 (.6) |
| CSD | **97.8** (0.) | **90.8** (.3) |

Table 4: In-domain and out-domain accuracies on rotated MNIST without batch augmentations. Shown are average and standard deviation from three runs.
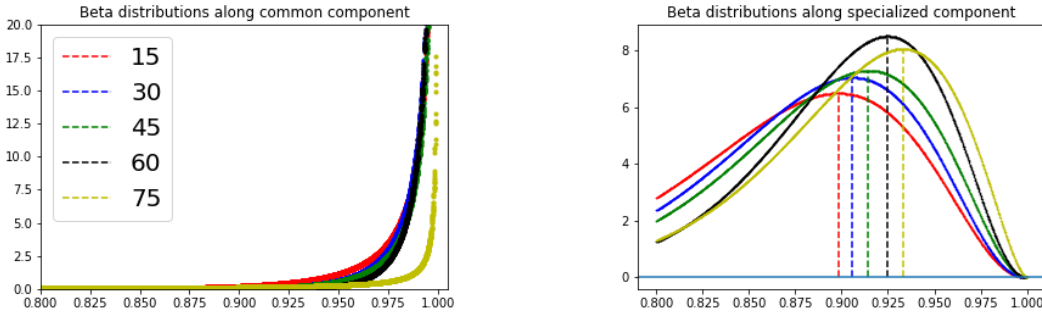
---

[4] https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html
[5] https://github.com/tensorflow/tensorflow/tree/r1.15/tensorflow/examples/speech_commands

**Rotated MNIST and Fashion-MNIST:** Rotated MNIST is a popular benchmark for evaluating domain generalization where the angle by which images are rotated is the proxy for domain. We randomly select[6] a subset of 2000 images for MNIST and 10,000 images for Fashion MNIST, the original set of images is considered to have rotated by $0°$ and is denoted as $\mathcal{M}_0$. Each of the images in the data split when rotated by $\theta$ degrees is denoted $\mathcal{M}_\theta$. The training data is union of all images rotated by $15°$ through $75°$ in intervals of $15°$, creating a total of 5 domains. We evaluate on $\mathcal{M}_0, \mathcal{M}_{90}$. In that sense only in this artificially created domains, are we truly sure of the test domains being outside the span of train domains. Further, we employ batch augmentations such as flip left-right and random crop since they significantly improve generalization error and are commonly used in practice. We train using the ResNet-18 architecture.

Table 3 compares the baseline, MASF, and CSD on MNIST and Fashion-MNIST. We show accuracy on test set from the same domains as training (in-domain) and test set from $0°$ and $90°$ that are outside the training domains. Note how the CSD's improvement on in-domain accuracy is insignificant, while gaining substantially on out of domain data. This shows that CSD specifically targets domain generalization. Surprisingly MASF does not perform well at all, and is significantly worse than even the baseline. One possibility could be that the domain-invariance loss introduced by MASF conflicts with the standard data augmentations used on this dataset. To test this, we compared all the methods without such augmentation. We observe that although all numbers have dropped 1–4%, now MASF is showing sane improvements over baseline, but CSD is better than MASF even in this setting.

## 4.1 How does CSD work?



(a) Beta fit on estimated probabilities of correct class using common common component.

(b) Beta fit on estimated probabilities of correct class using specialized component.

Figure 1: Distribution of probability assigned to the correct class using common or specialized components alone.

We provide empirical evidence in this section that CSD effectively decomposes common and low-rank specialized components. Consider the rotated MNIST task trained on ResNet-18 as discussed in Section 4. Since each domain differs only in the amount of rotation, we expect $W_s$ to be of rank 1 and so we chose $k = 1$ giving us one common and one specialized component. We are interested in finding out if the common component is agnostic to the domains and see how the specialized component varies across domains.

We look at the probability assigned to the correct class for all the train instances using only common component $w_c$ and using only specialized component $W_s$. For probabilities assigned to examples in each domain using each component, we fit a Beta distribution. Shown in Figure 1(a) is fitted beta distribution on probability assigned using $w_c$ and Figure 1(b) for $w_s$. Note how in Figure 1(a), the colors are distinguishable, yet are largely overlapping. However in Figure 1(b), notice how modes corresponding to each domain are widely spaced, moreover the order of modes and spacing between them cleanly reflects the underlying degree of rotation from $15°$ to $75°$.

These observations support our claims on utility of CSD for low-rank decomposition.

---

[6] The earlier work on this dataset however lacks standardization of splits, train sizes, and baseline network across the various papers Shankar et al. (2018) Wang et al. (2019). Hence we rerun experiments using different methods on our split and baseline network.

## 4.2 Ablation study

In this section we study the importance of each of the three terms in CSD's final loss: common loss computed from $w_c$ ($\mathcal{L}_c$), specialized loss ($\mathcal{L}_s$) computed from $w_i$ that sums common ($w_c$) and domain-specific parameters ($W_s, \Gamma$), orthonormal loss ($\mathcal{R}$) that makes $w_c$ orthogonal to domain specific softmax (Refer: Algorithm 1). In Table 5, we demonstrate the contribution of each term to CSD loss by comparing accuracy on LipitK with 76 domains.

| Common loss $\mathcal{L}_c$ | Specialized loss $\mathcal{L}_s$ | Orthonormality regularizer $\mathcal{R}$ | Accuracy |
|---|---|---|---|
| Y | N | N | 85.5 (.7) |
| N | Y | N | 84.4 (.2) |
| N | Y | Y | 85.3 (.1) |
| Y | N | Y | 85.7 (.4) |
| Y | Y | N | 85.8 (.6) |
| Y | Y | Y | 87.3 (.3) |

Table 5: Ablation analysis on CSD loss using LipitK (76)

The first row is the baseline with only the common loss. The second row shows prior decomposition methods that imposed only the specialized loss without any orthogonality or a separate common loss. This is worse than even the baseline. This can be attributed to decomposition without identifiability guarantees thereby losing part of $w_c$ when the specialized $W_s$ is discarded. Using orthogonal constraint, third row, fixes this ill-posed decomposition however, understandably, just fixing the last layer does not gain big over baseline. Using both common and specialized loss even without orthogonal constraint showed some merit, perhaps because feature sharing from common loss covered up for bad decomposition. Finally, fixing this bad decomposition with orthogonality constraint and using both common and specialized loss constitutes our CSD algorithm and is significantly better than any other variant.

This empirical study goes on to show that both $\mathcal{L}_c$ and $\mathcal{R}$ are important. Imposing $\mathcal{L}_c$ with $w_c$ does not help feature sharing if it is not devoid of specialized components from bad decomposition. A good decomposition on final layer without $\mathcal{L}_c$ does not help generalize much.

## 4.3 Importance of Low-Rank

Table 6 shows accuracy on various speech tasks with increasing k controlling the rank of the domain-specific component. Rank-0 corresponds to the baseline ERM without any domain-specific part. We observe that accuracy drops with increasing rank beyond 1 and the best is with $k = 1$ when number of domains $D \leq 100$. As we increase D to 200 domains, a higher rank (4) becomes optimal and the results stay stable for a large range of rank values. This matches our analytical understanding resulting from Theorem 1 that we will be able to successfully disentangle only those domain specific components which have been observed in the training domains, and using a higher rank will increase noise in the estimation of $w_c$.

| Rank $k$ | 50 | 100 | 200 |
|---|---|---|---|
| 0 | 72.6 (.1) | 80.0 (.1) | 86.8 (.3) |
| 1 | **74.1** (.3) | **81.4** (.4) | 87.3 (.5) |
| 4 | 73.7 (.1) | 80.6 (.7) | **87.5** (.1) |
| 9 | 73.0 (.6) | 80.1 (.5) | **87.5** (.2) |
| 24 | 72.3 (.2) | 80.5 (.4) | 87.4 (.3) |

Table 6: Effect of rank constraint (k) on test accuracy for Speech task with varying number of train domains.

## 5 Conclusion

We considered a natural multi-domain setting and looked at how standard classifier could overfit on domain signals and delved on efficacy of several other existing solutions to the domain generalization problem. Motivated by this simple setting, we developed a new algorithm called CSD that effectively decomposes classifier parameters into a common part and a low-rank domain-specific part. We presented a principled analysis to provide identifiability results of CSD while delineating the underlying assumptions. We analytically studied the effect of rank in trading off domain-specific noise suppression and domain generalization, which in earlier work was largely heuristics-driven.

We empirically evaluated CSD against four existing algorithms on five datasets spanning speech and images and a large range of domains. We show that CSD generalizes better and is considerably faster than existing algorithms, while being very simple to implement. In future, we plan to investigate algorithms that combine data augmentation with parameter decomposition to gain even higher accuracy on test domains that are related to training domains.

# References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pp. 998–1008, 2018.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. Analysis of representations for domain adaptation. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, 2006. URL http://dl.acm.org/citation.cfm?id=2976456.2976474.

Carlucci, F. M., D'Innocente, A., Bucci, S., Caputo, B., and Tommasi, T. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

DauméIII, H. Frustratingly easy domain adaptation. pp. 256–263, 2007.

DauméIII, H., Kumar, A., and Saha, A. Co-regularization based semi-supervised domain adaptation. In *NIPS*, pp. 478–486, 2010.

Dou, Q., de Castro, D. C., Kamnitsas, K., and Glocker, B. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pp. 6447–6458, 2019.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Ghifary, M., Bastiaan Kleijn, W., Zhang, M., and Balduzzi, D. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, pp. 2551–2559, 2015.

Hoffman, J., Mohri, M., and Zhang, N. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pp. 8246–8256, 2018.

Khosla, A., Zhou, T., Malisiewicz, T., Efros, A., and Torralba, A. Undoing the damage of dataset bias. In *ECCV*, pp. 158–171, 2012.

Li, D., Yang, Y., Song, Y., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision, ICCV*, 2017.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. *CVPR*, 2018b.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pp. 1041–1048, 2009.

Motiian, S., Piccirilli, M., Adjeroh, D. A., and Doretto, G. Unified deep supervised domain adaptation and generalization. In *ICCV*, pp. 5715–5725, 2017.

Muandet, K., Balduzzi, D., and Schlkopf, B. Domain generalization via invariant feature representation. In *ICML*, pp. 10–18, 2013.

Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., and Sarawagi, S. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Dx7fbCW.

Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, pp. 5334–5344, 2018.

Wang, H., He, Z., Lipton, Z. C., and Xing, E. P. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.

# Efficient Domain Generalization via Common-Specific Low-Rank Decomposition
## (Appendix)

## 6 Evaluation on PACS dataset

PACS[7] is a popular domain generalization benchmark. The dataset in aggregate contains around 10,000 images from seven object categories collected from 4 different sources: Photo, Art, Cartoon and Sketch. Evaluation using this dataset trains on three of four sources and tests on left out domain. This setting is challenging since it tests generalization to a radically different target domain. Despite being a very popular dataset, evaluations using this dataset is laced with several inconsistent or unfair comparisons. Carlucci et al. (2019) uses ten percent of train data for validation (albeit different from the official split), while Dou et al. (2019), Balaji et al. (2018) do not use any validation split. Carlucci et al. (2019), Dou et al. (2019) use data augmentation techniques while Balaji et al. (2018) do not. Also, since the dataset is small and target domain is very far from source domains, the results are sensitive to optimization parameters such as learning rate, optimizer, learning rate schedule. As a result, the comparisons using this dataset across different implementations are rendered useless. Table 1 of Carlucci et al. (2019) highlights these differences even in baseline across different implementations; With such widely differing baseline numbers it is hard to compare across different methods. For this reason, we relegate evaluation of CSD on this dataset to supplementary material accompanied by this word of caution.

We present comparisons in Table 7 of our method with Carlucci et al. (2019) and baseline while using their implementation for a fair comparison. The baseline network is ResNet-18. We perform almost the same or slightly better than image specific JiGen Carlucci et al. (2019). We stay away from comparing with Dou et al. (2019) since their reported numbers from different implementation, which is unavailable for ResNet-18, could have different baseline number.

| Method | Photo | Art | Cartoon | Sketch | Average |
|--------|-------|-------|---------|--------|---------|
| ERM | 95.73 | 77.85 | 74.86 | 67.74 | 79.05 |
| JiGen | 96.03 | 79.42 | 75.25 | 71.35 | 80.41 |
| CSD | 95.45 | 79.79 | 75.04 | 72.46 | **80.69** |

Table 7: Comparison between JiGen Carlucci et al. (2019) and CSD (ours) using PACS datset with ResNet-18 architecture. The header of each column identifies the target domain.

## 7 Proof of Theorem 1

*Proof of Theorem 1.* The high level outline of the proof is to show that the first two steps obtain the best rank-$(k+1)$ approximation of $W$ such that the row space contains $\mathbb{1}$. The last two steps retain this low rank approximation while ensuring that $w_c^{\text{new}} \perp \text{Span}\left(W_s^{\text{new}}\right)$.

The proof that the first two steps obtain the best rank-$(k+1)$ approximation follows that of the classical low rank approximation theorem of Eckart-Young-Mirsky. We first note that the minimization problem of $f$ under the given constraints, can be equivalently written as:

$$\min_{\widetilde{W}} \left\| W - \widetilde{W} \right\|_F^2$$
$$\text{s.t. rank}(\widetilde{W}) \leq k + 1 \text{ and } \mathbb{1} \in \text{Span}\left(\widetilde{W}^\top\right). \tag{7}$$

Let $\widetilde{w} := \frac{1}{D} W \mathbb{1}$ and $W - \widetilde{w}\mathbb{1}^\top = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^\top$ be the SVD of $W - \widetilde{w}\mathbb{1}^\top$. Since $\left(W - \widetilde{w}\mathbb{1}^\top\right)\mathbb{1} = 0$, we have that $\mathbb{1} \perp \text{Span}\left(\widetilde{V}\right)$. We will first show that $\widetilde{W}^* := \widetilde{w}\mathbb{1}^\top + \widetilde{U}_k \widetilde{\Sigma}_k \widetilde{V}_k^\top$, where $\widetilde{U}_k \widetilde{\Sigma}_k \widetilde{V}_k^\top$ is the top-$k$ component of $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^\top$, minimizes both $\left\| W - \widetilde{W} \right\|_F^2$ and $\left\| W - \widetilde{W} \right\|_2^2$ among all matrices $\widetilde{W}$ satisfying the conditions in (7). Let $\sigma_i$ denote the $i^{\text{th}}$ largest singular value of $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^\top$.

**Optimality in operator norm**: Fix any $\widetilde{W}$ satisfying the conditions in (7). Since $\mathbb{1} \in \text{Span}\left(\widetilde{W}^\top\right)$ and rank $\left(\widetilde{W}\right) = k + 1$, there is a unit vector $\widetilde{v} \in \text{Span}\left(\widetilde{V}_{k+1}\right)$ such that $\widetilde{v} \perp \text{Span}\left(\widetilde{W}^\top\right)$. Let $\widetilde{v} = \widetilde{V}_{k+1}x$. Since $\|\widetilde{v}\| = 1$ and $\widetilde{V}$ is an

---

[7] https://domaingeneralization.github.io/

orthonormal matrix, we have $\sum_i x_i^2 = 1$. We have:

$$\left\| W - \widetilde{W} \right\|_2^2 \geq \left\| (W - \widetilde{W})\widetilde{v} \right\|^2 = \|W\widetilde{v}\|^2 = \left\| W\widetilde{V}_{k+1}x \right\|^2$$

$$= \left\| \widetilde{U}_{k+1}\widetilde{\Sigma}_{k+1}x \right\|^2 = \sum_{i=1}^{k+1} \widetilde{\sigma}_i^2 \cdot x_i^2 \geq \widetilde{\sigma}_{k+1}^2$$

$$= \left\| W - \widetilde{w}\mathbb{1}^\top \right\|^2.$$

This proves the optimality in operator norm.

**Optimality in Frobenius norm**: Fix again any $\widetilde{W}$ satisfying the conditions in (7). Let $\sigma_i(A)$ denote the $i^{\text{th}}$ largest singular value of $A$ and $A_i$ denote the best rank-$i$ approximation to $A$. Let $W' := W - \widetilde{W}$. We have that:

$$\sigma_i(W') = \left\| W' - W'_{i-1} \right\| = \left\| W' - W'_{i-1} \right\| + \left\| \widetilde{W} - \widetilde{W} \right\|$$

$$\leq \left\| W' + \widetilde{W} - W'_{i-1} - \widetilde{W} \right\|$$

$$= \left\| W - W'_{i-1} - \widetilde{W} \right\|$$

$$\geq \min_{\widehat{W}} \left\| W - \widehat{W} \right\|,$$

where the minimum is taken over all $\widehat{W}$ such that $\text{rank}\left(\widehat{W}\right) \leq i + k$, $\mathbb{1} \in \text{Span}\left(\widehat{W}^\top\right)$. Picking $\widehat{W} = \widetilde{w}\mathbb{1}^\top + \widetilde{U}_{k+i-1}\widetilde{\Sigma}_{k+i-1}\widetilde{V}_{k+i-1}^\top$, we see that $\sigma_i(W - \widetilde{W}) \geq \sigma_{k+i}$. It follows from this that $\left\| W - \widetilde{W} \right\|_F \geq \left\| W - \widetilde{w}\mathbb{1}^\top - \widetilde{U}_k\widetilde{\Sigma}_k\widetilde{V}_k^\top \right\|_F$.

For the last two steps, note that they preserve the matrix $w_c\mathbb{1}^\top + W_s\Gamma^\top$. If $\overline{w}_c\mathbb{1}^\top + \overline{W}_s\overline{\Gamma}^\top$ is the unique way to write $w_c\mathbb{1}^\top + W_s\Gamma^\top$ such that $\overline{w}_c \perp \text{Span}\left(\overline{W}_s\right)$, then we see that $\overline{w}_c^\top \left(\overline{w}_c\mathbb{1}^\top + \overline{W}_s\overline{\Gamma}^\top\right) = \|\overline{w}_c\|^2 \mathbb{1}^\top$, meaning that $\overline{w}_c^\top = \|\overline{w}_c\|^2 \left(w_c\mathbb{1}^\top + W_s\Gamma^\top\right)^+ \mathbb{1}$. This proves the theorem.

$\square$