Principal Sensitivity Analysis and Its Application to Knowledge Discovery in Functional Neuroimaging

Sotetsu Koyamada

Graduate School of Informatics, Kyoto University

In this thesis, we studied methods to visualize the knowledge captured by supervised classifiers. In particular, we developed a new method, "Principal Sensitivity Analysis (PSA)," to analyze the sensitivity of the trained classifier. In PSA, principal sensitivity map (PSM) is defined as the direction in the input space to which the classifier is most sensitive, and k-th PSM is also analogously defined for each k. Using these maps, PSA decomposes the input space based on the sensitivity of the classifier. As a primitive case study, we first applied the PSA to the classifier trained for digit classification. We were able to find a direct association between the PSMs and the discriminative features of the digits that we humans intuitively use for classification. Next, in order to assess the performance of our algorithm on nonlinear and hierarchical classifiers in a practical setting, we applied the PSA to the deep neural network (DNN) trained with large-scale neuroimaging database. We confirmed that, in comparison to other baseline methods, the DNN can capture richer discriminative features of brain activities that are common to many human subjects. Interestingly, we were able to find nontrivial connections between the PSMs of the trained DNN and the functional connectivity among brain areas, a phenomenon studied in neuroscience. This suggests a relation between the discriminative features of brain activities and the functional connectivity.

Contents

1 Introduction

Machine learning is a powerful methodology to construct efficient and robust predictors and classifiers. In particular, deep learning, a technology to train complicated neural networks with multiple hidden layers, i.e., deep neural networks (DNNs), has been attracted much attention because of its outstanding performance in several artificial intelligence tasks [1, 2, 3]. Now, literature suggests the ability of supervised learning not only to reproduce "intuition and experience" based on human supervision [3], but also to successfully classify the objects that humans cannot sufficiently classify with inspection alone [4, 5]. This work is motivated by the cases in which the supervised classifier eclipses the human decisions.

In such cases, one might be able to learn the knowledge of the data from the trained classifiers. We may say that the case that the classifiers defeat the human occurs when the classifier holds more knowledge about the data and the subject classes than us, because our incompetence in the classification problems can be attributed solely to our lack of understanding about the class properties and/or the similarity metrics. For example, Miyawaki et al. [6] used logistic regression for this purpose in neuroimaging. Let us consider the classifier $f : \mathbb{R}^d \to [0, 1]$ with the d-dimensional input $\boldsymbol{x}$. Logistic regression is a linear model

$$f(\boldsymbol{x}) = \sigma\left(\sum_i^d w_i x_i + b\right), \tag{1}$$

where w_i and b denote a weight parameter and a bias parameter, respectively, and σ is a sigmoid function. Miyawaki et al. assigned each small three-dimensional brain unit (voxel) to a input dimension, and visualized the determined weights, $(w_1, \ldots, w_d)$ as brain map in order to extract the knowledge captured by the trained classifier. One can understand that if the absolute value of weight $|w_i|$ is large enough, then the corresponding i-th input dimension (voxel) is important for characterizing the subject class. Other neuroimaging studies [7, 8, 9] also provided such visualization of linear models.

Unfortunately, however, trained classifiers are often so complex that they defy the human interpretation. If one add a hidden layer to the logistic regression:

$$f(\boldsymbol{x}) = \sigma\left(\sum_i^l w_i^{(2)} h\left(\sum_j^d w_{ij}^{(1)} x_j + b_i^{(1)}\right) + b^{(2)}\right), \tag{2}$$

where h is a nonlinear activation function, then the nonlinear classifier f becomes too complex for the human to directly analyze the relationship between the input data and the classes, because the hierarchical architecture of the classifier f masks the direct connection between the input and output. The equation (2) is the simplest version of feed-forward neural networks, and thus DNNs with more hidden layers are too complicated for the human to interpret their decision. However, the superiority of nonlinear machine learning techniques like deep learning [1, 2, 3] strongly suggests that the trained classifiers capture the "invisible" properties of the subject classes. Geoff Hinton solidified this into the philosophy of "dark knowledge" captured within the trained neural

networks [10]. Thus, our objective is to visualize the decision of such nonlinear and hierarchical machines like DNN in the interpretable form.

For the visualization of high-dimensional feature space of nonlinear machines, Zurada et al. [11, 12] and Kjems et al. [13] presented seminal works. Zurada et al. defined *sensitivity* by

$$s_i := E_q \left[\left(\frac{\partial f(\boldsymbol{x})}{\partial x_i} \right)^2 \right], \tag{3}$$

where q is the true distribution of $\boldsymbol{x}$, in order to "delete unimportant data components for feed-forward neural networks." Kjems et al. visualized Zurada's idea as *sensitivity map*, $(s_1, \ldots, s_d)$ in the context of neuroimaging. Note that when the classifier f is a linear model like logistic regression, the sensitivity satisfies $s_i \propto w_i^2$. Rasmussen et al. [14] applied this *sensitivity analysis* to a nonlinear kernel method in functional neuroimaging. Because the sensitivity analysis provides a single map by the integration (3) over the whole input space $\mathbb{R}^d$, we can regard the sensitivity map as a "global" visualization of the classifier f. Baehrens et al. [15] took another approach. Baehrens et al. defined *local explanation vector* by the gradient $\nabla f(\boldsymbol{x})$ and used it for analyzing the importance of features at each "localized" point in the input space. Although these efforts have been made to visualize the classifiers, there is still much room left for improvement. Machine learning techniques in neuroimaging, for example, prefer linear kernels to nonlinear kernels because of the lack of visualization techniques [7].

In this study, we attempt to generalize the idea of sensitivity analysis, and develop a new framework that aids us in extracting the knowledge from classifiers that are trained in a supervised manner. Our framework is superior to the predecessors in that it can:

1. be used to identify a pair of discriminative input features that act oppositely in characterizing a class,

2. identify *combinations* of discriminative features that strongly characterize the subject classes,

3. provide platform for developing sparse, visually intuitive sensitivity maps.

The new framework gives rise to the algorithm that we refer to as "Principal Sensitivity Analysis (PSA)," which is analogous to the well-established "Principal Component Analysis (PCA)." To demonstrate the PSA's ability to extract the knowledge captured by the trained classifiers, we applied the PSA to the classifier trained for digit classification.

As a practical application of PSA, we next used the PSA for functional neuroimaging. In order to capture the discriminative features of brain activities common to all human subjects, we trained a DNN with a large-scale neuroimaging database and confirmed that, in comparison to other baseline methods, the DNN captured more discriminative features. We then applied the PSA to the DNN-based decoder to highlight the subject-independent features used by the decoders for classifying brain activities. By illustrating these features as brain maps, we found the association between the features and functional connectivity established in human fMRI studies [16, 17, 18, 19].

This thesis is structured as follows. In Section 2, we will provide the PSA's algorithm along with the explanation of the conventional sensitivity analysis. In Section 3, we will explain the architecture of feed-forward neural networks, to which we apply the PSA, and also explain its training. In Section 4, we will show a primitive application of the PSA to the classifier trained for digit classification, and demonstrate its advantages in comparison to the conventional sensitivity analysis. In Section 5, we will also provide an practical application of the PSA to functional neuroimaging. Finally, in Section 6, we will summarize the results and make conclusive remarks.

2 Principal Sensitivity Analysis (PSA)

2.1 Conventional sensitivity analysis

Before introducing the PSA, we describe the original sensitivity map introduced in [13]. Let d be the dimension of the input space, and let $f : \mathbb{R}^d \to \mathbb{R}$ be the classifier function obtained from supervised training. In the case of SVM, f may be the discriminant function. In the case of nonlinear neural networks, f may represent the function (or log of the function) that maps the input to the output of a unit in the final layer. We are interested in the expected sensitivity of f with respect to the i-th input feature. This can be written as

$$s_i := E_q \left[\left(\frac{\partial f(\boldsymbol{x})}{\partial x_i} \right)^2 \right], \tag{4}$$

where q is the distribution over the input space. In actual implementation, the integral (4) is computed with the empirical distribution q of the test dataset. Now, the vector

$$\boldsymbol{s} := (s_1, \ldots, s_d) \tag{5}$$

of these values will give us an intuitive measure for the degree of importance that the classifier attaches to each input feature. Kjems et al. [13] defined $\boldsymbol{s}$ as **sensitivity map** over the set of input features.

2.2 Sensitivity in arbitrary direction

Here, we generalize the definition (4). We define $s(\boldsymbol{v})$ as the sensitivity of f in arbitrary direction $\boldsymbol{v} := \sum_i^d v_i \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ denotes the i-th standard basis in $\mathbb{R}^d$:

$$s(\boldsymbol{v}) := E_q \left[\left(\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{v}} \right)^2 \right]. \tag{6}$$

Recall that the directional derivative is defined by

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{v}} := \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{v}. \tag{7}$$

Note that when we define the *sensitivity kernel metric*

$$\boldsymbol{K} := E_q \left[\nabla f(\boldsymbol{x}) \nabla f(\boldsymbol{x})^{\mathrm{T}} \right], \tag{8}$$

we can rewrite $s(\boldsymbol{v})$ by

$$s(\boldsymbol{v}) = \boldsymbol{v}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{v}. \tag{9}$$

2.3 Formulation of PSA

The classical setting (5) was developed in order to quantify the sensitivity of f with respect to each individual input feature independently. We attempt to generalize this idea and seek the *combination of the input features* for which f is most sensitive, or the combination of the input features that is *"principal"* in the evaluation of the sensitivity of f. We can quantify such combination by the vector $\boldsymbol{v}$, solving the following optimization problem about $\boldsymbol{v}$:

$$\begin{aligned} &\text{maximize} \quad \boldsymbol{v}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{v} \\ &\text{subject to} \quad \boldsymbol{v}^{\mathrm{T}} \boldsymbol{v} = 1. \end{aligned} \tag{10}$$

The solution to this problem is simply the maximal eigenvector $\pm \boldsymbol{v}^*$ of $\boldsymbol{K}$. Note that v_i represents the contribution of the i-th input feature to this principal combination, and this gives rise to the map over the set of all input features. As such, we can say that $\boldsymbol{v}$ is the **principal sensitivity map (PSM)** over the set of input features. From now on, we call $\boldsymbol{s}$ in the classical definition (5) as the **standard sensitivity map** and make the distinction. The magnitude of v_i represents the extent to which f is sensitive to the i-th input feature, and the sign of v_i will tell us the relative direction to which the input feature influences f. The new map is thus richer in information than the standard sensitivity map. In Section 4.2 we will demonstrate the benefit of this extra information.

2.3.1 PSA

We can naturally extend our construction above and also consider other eigenvectors of $\boldsymbol{K}$. We can find these vectors by solving the following optimization problem about $\boldsymbol{V}$:

$$\begin{aligned} &\text{maximize} \quad \mathrm{Tr}\left(\boldsymbol{V}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{V}\right) \\ &\text{subject to} \quad \boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{v}_j = \delta_{ij}, \end{aligned} \tag{11}$$

where $\boldsymbol{V}$ is a $d \times d$ matrix. As is well known, such $\boldsymbol{V}$ is given by the invertible matrix with each column corresponding to $\boldsymbol{K}$'s eigenvector. We may define k-th dominant eigenvector $\boldsymbol{v}_k$ as the **k-th principal sensitivity map.** These sub-principal sensitivity maps grant us access to even richer information that underlies the dataset. We will show the benefits of these additional maps in Fig. 3. From now on, we will refer to the first PSM by just PSM, unless noted otherwise.

Recall that, in the ordinary PCA, $\boldsymbol{K}$ in the definition (11) is given by the covariance $E_q\left[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right]$, where $\boldsymbol{x}$ is the centered random variable. Thus, our algorithm can be seen as the PCA applied to the covariance of $\nabla f(\boldsymbol{x})$ without centering.

2.3.2 Sparse PSA

One may use the new definition (11) as a starting point to develop sparse, visually intuitive sensitivity maps. For example, we may introduce the existing techniques in sparse PCA and sparse coding into our framework. We may do so [20] by replacing

the covariance matrix in its derivation with our $\boldsymbol{K}$. In particular, we can define an alternative optimization problem about $\boldsymbol{V}$ and $\boldsymbol{\alpha}_i$:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\sum_{i}^{N} \|\nabla f(\boldsymbol{x}_i) - \boldsymbol{V}\boldsymbol{\alpha}_i\|_2^2 + \lambda \sum_{k}^{r} \|\boldsymbol{v}_k\|_1 \\ &\text{subject to} \quad \|\boldsymbol{\alpha}_i\|_2 = 1, \end{aligned} \tag{12}$$

where r is the number of sensitivity maps and N is the number of samples. For the implementation, we used scikit-learn [21].

3 Neural networks for classification

The classifiers to which we applied PSA in this study were feed-forward neural networks [22]. Thus, in this section, we will explain the architecture of feed-forward neural networks and training of them.

3.1 Architecture of feed-forward neural networks

We trained feed-forward neural networks with L hidden layers. The internal potential of the i-th unit in the l-th hidden layer $a_i^{(l)}$ $(l = 1, \cdots, L)$ is given as a weighted summation of its inputs:

$$a_i^{(l)} = \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} z_j^{(l-1)} + b_i^{(l)}, \tag{13}$$

where $w_{ij}^{(l)}$ and $b_i^{(l)}$ are a weight and a bias, respectively. Here, n_l is the number of units in the l-th hidden layer. We define $\boldsymbol{z}^{(0)}$ as the input vector $\boldsymbol{x}$ to the network. This forces n_0 to be equal to the input's dimensionality d. We denote $\boldsymbol{z}^{(l)} = \left(z_1^{(l)}, \cdots, z_{n_l}^{(l)}\right)$ as the outputs of the l-th hidden layer. These outputs are given by applying a nonlinear activation function h to the internal potential as

$$z_i^{(l)} = h(a_i^{(l)}). \tag{14}$$

As the activation function h, we used a sigmoid function $1/\left(1 + \exp(-x)\right)$ or ReLU [23], a piecewise linear function $max(0, x)$. ReLU as a choice of the activation function has a couple of advantages; its piecewise linearity can save the computational cost to calculate its derivative, and its non-saturating character in the positive domain prevents the learning algorithm from halting due to gradient vanishing of nonlinear activation functions. The last hidden layer was connected to the softmax (output) layer, so that the output from the k-th unit of the output layer can be interpreted as the posterior probability of class k, given by

$$P(Y = k \,|\, \boldsymbol{x}, \boldsymbol{W}) = \frac{\exp\left(\sum_{j=1}^{n_L} w_{kj} z_j^{(L)} + b_k\right)}{\sum_{k'=1}^{K} \exp\left(\sum_{j=1}^{n_L} w_{k'j} z_j^{(L)} + b_{k'}\right)}, \tag{15}$$

where K is the number of classes, and $\boldsymbol{W}$ denotes all the parameters (weights and biases) of the whole network. Here, Y is a random variable signifying the class to which $\boldsymbol{x}$ belongs.

3.2 Training via MSGD

We trained the neural network of the architecture above with stochastic gradient descent (SGD). We used a negative log-likelihood as the cost function of the learning

$$L(\boldsymbol{W}) = -\sum_{n=1}^{N} \log P(Y_n = t_n \,|\, \boldsymbol{x}_n, \boldsymbol{W}), \tag{16}$$

where $\{(\boldsymbol{x}_1, t_1), \cdots, (\boldsymbol{x}_N, t_N)\}$ constituted the given dataset. Here $t \in \{1, \cdots, K\}$ denotes a class label. In particular, to minimize the above cost function, minibatch stochastic gradient descent (MSGD) was introduced so that SGD was performed every 100 samples:

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \eta_t \left. \frac{\partial L'(\boldsymbol{W})}{\partial \boldsymbol{W}} \right|_{\boldsymbol{W}_{t-1}}, \tag{17}$$

where L' is the cost function for the cached subset of 100 samples in the minibatch, and η_t is the learning rate. Each weight of hidden layers was initialized at a small value randomly sampled from a zero-mean normal distribution with the standard deviation of 0.01, and weights of softmax layer and biases were initialized to zero. Early stopping was also adopted; if the classification performance for the validation dataset did not increase for 100 learning epochs, then learning was terminated.

To further avoid over-fitting, we used the dropout technique [24] in some cases. During the training, the activity $z_i^{(l)}$ was randomly replaced by 0 with probability p. We set $p = 0.5$ for hidden units and 0.2 for inputs. This drop-out procedure plays a role of regularization. When testing the trained neural network, all the nodes were activated, and their weights were multiplied by $1 - p$. This is to make the mean activity level of each network element consistent between the training phase and the test phase.

4 Application to digit classification

In order to demonstrate the effectiveness of the PSA, we applied the analysis to the classifiers that we trained with artificial data and MNIST data.

4.1 Specification of experiments

Our artificial data is a simplified version of the MNIST data in which the object's *orientation* and *positioning* are registered from the beginning. All samples in the artificial data are constructed by adding noises to the common set of templates representing the numerics from 0 through 9 (see Fig. 1). We then fabricated the artificial noise in three steps: we (1) flipped the bit of each pixel in the template picture with probability $p = 0.2$, (2) added Gaussian noise $\mathcal{N}(0, 0.1)$ to the intensity, and (3) truncated the negative intensities. The sample size was set to be equal to that of MNIST. Our training data, validation data, and test data consisted respectively of 50,000, 10,000, and 10,000 sample patterns.

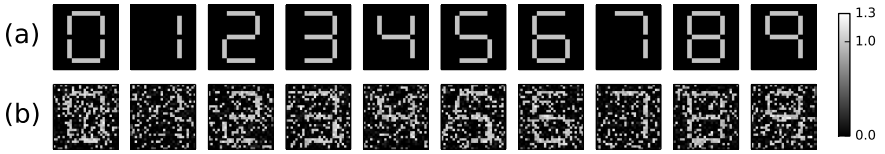

Figure 1: (a) Templates. (b) Noisy samples. Each figure is of 28×28 pixels.

Using the artificial dataset above and the standard MNIST, we trained a feed-forward neural network for the classification of ten numerics. In Table 1, we provide the structure of the neural network and its performance over each dataset. For either dataset, the training was conducted via MSGD with constant learning rate η. We also adopted a dropout method [24] only for the training on the MNIST dataset. The output from each unit in the final layer is given by the posterior probability of each class c. For computational purpose, we transform this output by log:

$$f_c(\boldsymbol{x}) := \log P(Y = c \,|\, \boldsymbol{x}), \tag{18}$$

We then constructed the PSMs and the standard sensitivity map for the f_c given above.

Table 1: Summary of training setups based on neural networks

Data set	**Architecture**	**Unit type**	**Dropout**	**Learning rate**	**Error[%]**
Digital data	784-500-10	Logistic	No	0.1	0.36
MNIST	784-500-500-10	ReLU	Yes	0.1	1.37

4.2 PSA of the classifier trained on artificial dataset

We will describe three ways in which the PSA can be superior to the analysis based on standard sensitivity map.

Fig. 2 compares the PSMs and standard sensitivity maps, which were both obtained from the common neural networks trained for the same 10-class classification problem. The color intensity of i-th pixel represents the magnitude of s_i or v_i. Both maps capture the characters that the "colorless" rims and likewise "colorless" regions enclosed by edges are insignificant in the classification. Note that the (first) PSM distinguishes the types of sensitivities by their sign. For each numeral, the PSM assigns opposite signs to "the edges whose *presence* is crucial in the characterization of the very numeral" and "the edges whose *absence* is crucial in the numeral's characterization." This information is not featured in the standard sensitivity map. For instance, in the sensitivity map for the numeral 1, the two edges on the right and the rest of the edges have the opposite sensitivity. As a result, we can verify the red figure of 1 in its PSM. We are able to clearly identify the unbroken figures of $2, 4, 5$ and 9 in their corresponding PSM as well. We see that, with the extra information regarding the sign of the sensitivity over each pixel, PSM can provide us with much richer information than the standard counterpart.

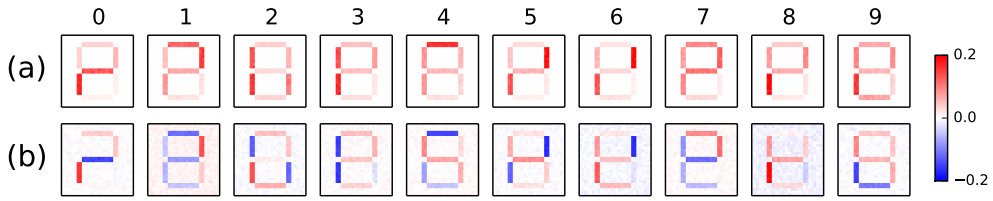

Figure 2: (a) Standard sensitivity maps. (b) PSMs.

Next, we will show the benefits of sub-principal sensitivity maps computed from PSA. Fig. 3(a) shows the first PSM through the third PSM for the numerals 0 and 9.[1] In order to show how this extra information benefits us in visualization of the classification problem, we consider the following "local" sensitivity map integrated over the samples from a particular pair of classes:

$$s_{c,c'}(\boldsymbol{v}) := E_{q_{c,c'}}\left[\left(\frac{\partial f_c(\boldsymbol{x})}{\partial \boldsymbol{v}}\right)^2\right], \tag{19}$$

where $q_{c,c'}$ is the empirical distribution over the set of samples generated from the classes c and c'. To get the intuition about this value, note that the value for $(c, c') = (9, 4)$ can also be pictorially written as

$$\lim_{\varepsilon\to 0} E_{\{9,4\}}\left[\left(\frac{\log P\left(Y = \boxed{9} \mid \boxed{} + \varepsilon\boldsymbol{v}\right) - \log P\left(Y = \boxed{9} \mid \boxed{}\right)}{\varepsilon}\right)^2\right], \tag{20}$$

[1] We list the PSMs for all the numerals $(0, \ldots, 9)$ in the Appendix A.

where $\boldsymbol{v}$ can be the third PSM of class 9,  , for example. If $\boldsymbol{v}_k$ is the k-th PSM of the classifier, then $s_{c,c'}(\boldsymbol{v}_k)$ quantifies *the sensitivity of the machine's answer to the binary classification problem of "c vs c′"* with respect to the perturbation of the input in the direction of $\boldsymbol{v}_k$. By looking at this value for each k, we may visualize the ways that the classifier deals with the binary classification problem. Such visualization may aid us in learning from the classifiers the way to distinguish one class from another. Fig. 3(b) shows the values of $s_{c,c'}(\boldsymbol{v}_k)$ for $c \in \{0, 9\}$ and $k \in \{1, \ldots, 10\}$. We could see in the figure that, for the case of $(c, c') = (9, 4)$, $s_{c,c'}(\boldsymbol{v}_3)$ was larger than $s_{c,c'}(\boldsymbol{v}_1)$. This suggests that the third PSM is more helpful than the first PSM for distinguishing 4 from 9. We can actually verify this fact by observing that the third PSM is especially intense at the top most edge, which can alone differentiate 4 from 9. We are able to confirm many other cases in which the sub-principal sensitivity maps were more helpful in capturing the characters in binary classification problems than the first PSM. Thus, PSA can provide us with the knowledge of the classifiers that was inaccessible with the previous method based on the standard sensitivity map.

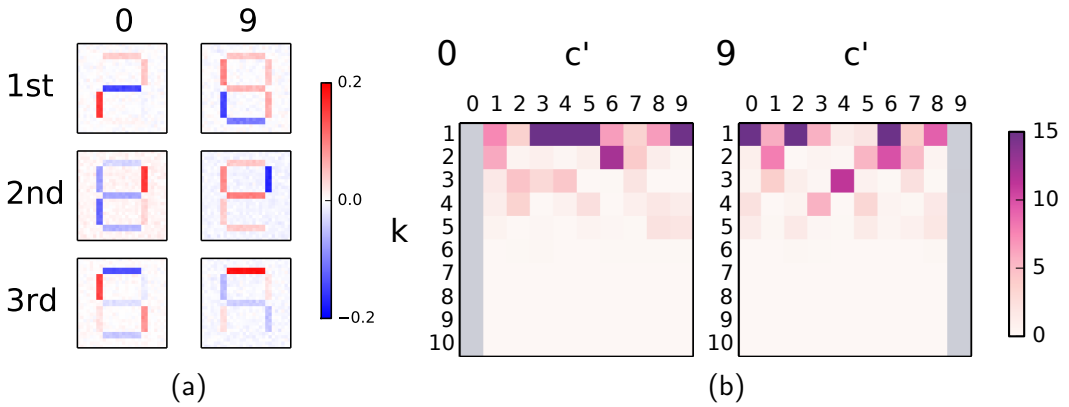

Figure 3: (a) First, second and third PSMs of the classifier outputs f_c for the numerals 0 and 9. (b) $s_{c,c'}(\boldsymbol{v}_k)$ for $c \in \{0, 9\}$, $k \in \{1, \ldots, 10\}$, and $c' \in \{0, \ldots, 9\} \backslash \{c\}$.

Finally, we demonstrate the usefulness of formulation (11) in the construction of sparse and intuitive sensitivity map. Fig. 4 depicts the sensitivity maps obtained from the application of our sparse PSA in (12) to the data above. Note that the sparse PSA not only washes away rather irrelevant pixels from the canvas, but it also assigns very high intensity to essential pixels. With these "localized" maps, we can better understand the discriminative features utilized by the trained classifiers.

4.3 PSA of the classifier trained on MNIST dataset

We trained a nonlinear neural network-based classifier on the MNIST dataset, which consists of hand-written digits from 0 through 9. We then analyzed the trained classifier with our PSA. This dataset illuminates a particular challenge to be confronted in the application of the PSA. By default, hand-written objects do not share common displace-

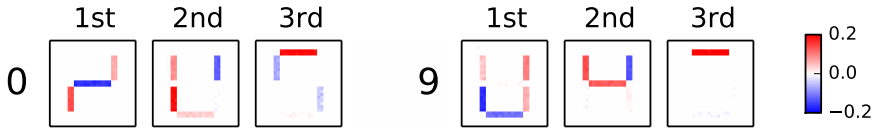

Figure 4: Results of the sparse PSA on the classifiers f_c with $r = 3$ for the numerals 0 and 9. We ranked the 3 basis elements by the magnitude of $s(\boldsymbol{v})$. We selected the regularization term of $\lambda = 5$, and each PSM was normalized so that its L_2 norm was 1.

ment and orientation. Without an appropriate registration of input space, the meaning of each pixel can vary across the samples, making the visualization unintuitive. This is typical in some of the real-world classification problems. In the fields of applied science, standard registration procedure is often applied to the dataset before the construction of the classifiers. For example, in neuroimaging, one partitions the image data into anatomical regions after registration based on the standard brain, and represents each one of them by a group of voxels. In other areas of science, one does not necessarily have to face such problems. In genetics, data can be naturally partitioned into genes [25]. Likewise, in meteorology, 3D dataset is often translated into voxel structures, and a group of voxels may represent geographical region of specific terrain [26]. In this light, the digit recognition in unregistered MNIST data may not be an appropriate example for showing the effectiveness of our visualization method. For the reason that we will explain later, registration of multiclass dataset like MNIST can be difficult. We chose MNIST dataset here because it is familiar in the community of machine learning.

Fig. 5 summarizes the results. Both the standard sensitivity map and the PSM were able to capture the character that outer rims are rather useless in the classification.

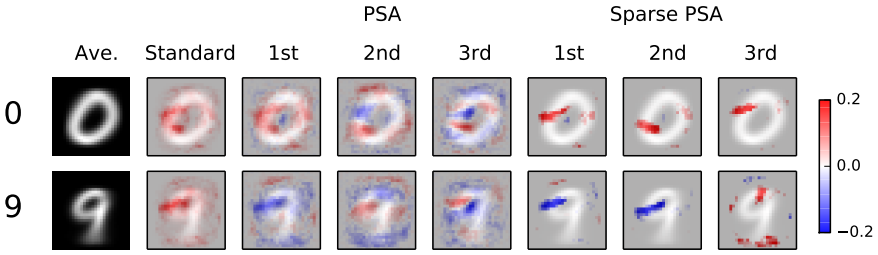

Figure 5: Standard sensitivity maps, PSMs, and sparse PSMs for $c \in \{0, 9\}$, $k \in \{1, 2, 3\}$, and $c' \in \{0, \ldots, 9\} \backslash \{c\}$. Ave. stands for the average of the testing dataset for the corresponding numerals.

Fig. 6 shows the values of $s_{c,c'}(\boldsymbol{v}_k)$. We can verify that small numbers of PSMs are complementing each other in their contributions to the binary classifications.

We also applied sparse PSA to the classifier with $r = 3$ and $\lambda = 40$ (see Fig. 5). We see that the sparse PSA highlights the essential pixels much more clearly than the normal

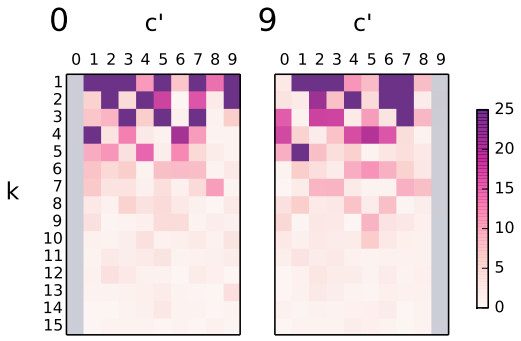

Figure 6: $s_{c,c'}(\boldsymbol{v}_k)$ for $c \in \{0, 9\}$, $k \in \{1, \ldots, 15\}$, and $c' \in \{0, \ldots, 9\} \backslash \{c\}$.

PSA.

Since the orientation and position of each numeral pattern varies across the samples in this dataset, input dimensions hold different meanings in different samples. To perform more effective visualization, we would need registration to adjust each numeral pattern to a common standard template. This problem might not be straightforward, since one must prepare different templates for different numeral patterns. An elegant standardization suitable for our PSA-based visualization remains as a future study.

5 Application to functional neuroimaging

The objective of this section is to apply the PSA to knowledge discovery in a practical setting and to assess its ability to illustrate the decision of the nonlinear and hierarchical classifier trained with real data. We applied the PSA to brain decoding. Because each input is aligned to the standard space (brain), brain decoding is an appropriate application of the PSA.

Brain decoding is an act of decoding exogenous and/or endogenous brain states from measurable brain activities [27, 28, 29, 30, 4]. It has been attracting much attention in medical and industrial fields as a major next-generation technology. Possible applications include brain machine interface (BMI) [31], neuro rehabilitation [32] and therapy of mental disorders [33]. Brain decoding is a function that takes brain activities as input and brain states as output, and its performance is evaluated by how well it approximates the real association between these two entities. As such, it falls into the category of machine learning, and it is often studied in the particular framework of supervised learning [34]. The brain decoder that well approximates the relationship between brain activities and brain states might have the knowledge about the brain. Thus, one might consider to extract and visualize the knowledge acquired by the decoder. We do this by applying the PSA to the efficient brain decoder.

In order to extract the features common to all human beings, we trained the decoder as subject-transfer decoder [35, 36, 37] by a large-scale functional magnetic resonance imaging (fMRI) database. For an ideal subject-transfer decoder, its decoding accuracy does not deteriorate over the dataset obtained from the population outside the group of individuals used in its training. To train a better subject-transfer decoder, we will present a subject-transfer decoder in the form of a neural network with multiple hidden layers (DNN), a decoder aimed at classifying the brain activities into seven cognitive task categories. For both training and testing, we used a large open fMRI dataset in Human Connectome Project (HCP) gathered from over 500 subjects [38]. In this section, we will show the efficiency of DNN-based decoder in subject-transfer decoding and thus in capturing the subject-independent features from big data.

Our subject-transfer decoder achieved higher decoding accuracy than any other baseline methods like support vector machine (SVM). This is indicative of DNN's superior generalization ability over heterogeneous big data. Also, decoding accuracy improved monotonically as the number of training subjects increased. In the light of the fact that we are engaged in subject-transfer decoding, this monotonic trend suggests that our training is successfully extracting more subject-independent features from larger dataset. In order to extract and visualize these features, we applied PSA to the trained subject-transfer decoder. By illustrating these features on the map of brain, we were able to find some connections between these features and functional connectivity reported in human fMRI studies [16, 17, 18, 19].

The following sections are structured as follows. In Section 5.1, 5.2 and 5.3, we will provide the specification of HCP dataset, the summary of our DNN-based decoder and the evaluation of the subject-transferability, respectively. In Section 5.4, we will compare the performance of our DNN against standard classification techniques, and

show that the decoding accuracy of the DNN improved as the size of the training dataset increased. In Section 5.5, the PSMs of the subject-transfer decoder and its interpretation are summarized.

5.1 fMRI data acquisition and preprocessing

Human Connectome Project (HCP) is a scientific project "to map macroscopic human brain circuits and their relationship to behavior in a large population of healthy adults" [38], and it provides one of the largest open databases of fMRI that are publically available today. Here, we used the task-evoked fMRI data collected from 499 participants in Quarter 1 through Quarter 6, which were preprocessed and registered by [38, 39] (HCP S500 release). For more details, see the HCP release reference manual[2].

In this section, we will provide key data specifications and preprocessing procedure. fMRI data of 499 healthy adults were acquired by a Siemens 3T Skyra, with TR = 720 ms, TE = 33.1 ms, flip angle 52°, FOV = 208×180 mm, 72 slices, 2.0×2.0 mm in plane resolution. The preprocessing that had been applied to the fMRI data in the HCP prior to our own modification includes removal of spatial artifacts and distortions, within-subject cross-modal registrations, reduction of the bias field, and alignment to standard space. In addition to these processes, we applied voxel-wise z-score transformation to the data and averaged the intensity over each anatomical region of interest (aROI). The intention of the latter averaging procedure is to help the decoder learn features that are robust against large inter-subject variability of brain activities. aROIs were determined by the automated anatomical labeling method [40]. In the end, the dimension per each preprocessed fMRI scan became 116.

The 499 participants (subjects) in the dataset we studied were asked to perform seven tasks related to the following categories: Emotion, Gambling, Language, Motor, Relational, Social and Working Memory (WM). Each subject performed each task twice with time limits that varies across different tasks (see Table 2). Note that the number of scans conducted in the experiment varies across different tasks. One hundred unrelated subjects completed all seven tasks. The WM class occupied the largest proportion (20.88%) of all scans for each subject. The experimental design of each task is summarized below. See [41] for more details.

Table 2: Number of scans per session and its duration (min:sec)

	Emotion	**Gambling**	**Language**	**Motor**	**Relational**	**Social**	**WM**
Scans	176	253	316	284	232	274	405
Duration	2:16	3:12	3:57	3:34	2:56	3:27	5:01

1. **Emotion:** Participants were asked to match one of two simultaneously presented images with a target image (angry face or fearful face). This is a modified version of the emotion task employed in [42].

[2] www.humanconnectome.org/documentation/S500

2. **Gambling:** Participants were asked to play a simple game to get money. See [43] for more details.

3. **Language:** After listening to a brief story, participants were asked to answer a two-alternative forced choice question about the topic of the story. See [44] for more details.

4. **Motor:** Participants were requested to move one of five body parts (left or right finger, left or right toe, or tongue) as instructed by a visual cue [45].

5. **Relational:** Each participant was presented with two pairs of objects, and was subsequently asked to answer a second-order question regarding the shapes/textures of the objects.

6. **Social:** Participants were presented with a movie clip, and were asked to decide whether the movements of the objects in the clips are related with each other in some way. The movie clips were originally prepared by [46] and [47].

7. **WM (Working Memory):** Participants were asked to complete two-back working memory tasks and zero-back tasks with four different types of image stimuli (places, tools, faces or body parts).

5.2 Deep neural networks as decoder

We trained a deep neural network (DNN) with the input being the fMRI signals over aROIs and the output being their labeled task classes, i.e., the category of cognitive task performed by the participants. Prior to the training step, all fMRI scans were categorized into seven task classes, completely disregarding the time order. The weight parameters of the DNN were then trained to optimize the probability of successfully classifying the fMRI scans into the seven task categories (Fig. 7).

The architecture of the trained DNNs are feed-forward neural networks (see Section 3) with one, two and three hidden layers ($n_l = 500$ for any $l > 0$). We utilized ReLU for all activation function. During training, we adopted a constant learning rate η, and this value was set at either of $\{0.1, 1.0\}$ that yielded better result for the validation dataset (see Section 5.3). In order to avoid over-fitting, we adopted the dropout technique [24]. Dropout is expected to prevent the decoder from acquiring subject-specific features.

5.3 Subject-transfer decoding

To examine the subject-transferability of our decoder's architecture, we selected hundred individuals from all 499 subjects who (1) are unrelated with each other and (2) successfully completed all seven cognitive tasks twice. Let D be the dataset of these 100 subjects. We then executed a *leave-10-subjects-out* (or *10-folds*) cross validation to the dataset D. To be more specific, the D was partitioned into a test dataset of 10 subjects, a validation dataset of 10 subjects, and a training dataset of 80 subjects without any overlap. We trained our DNN with the training dataset, while using the

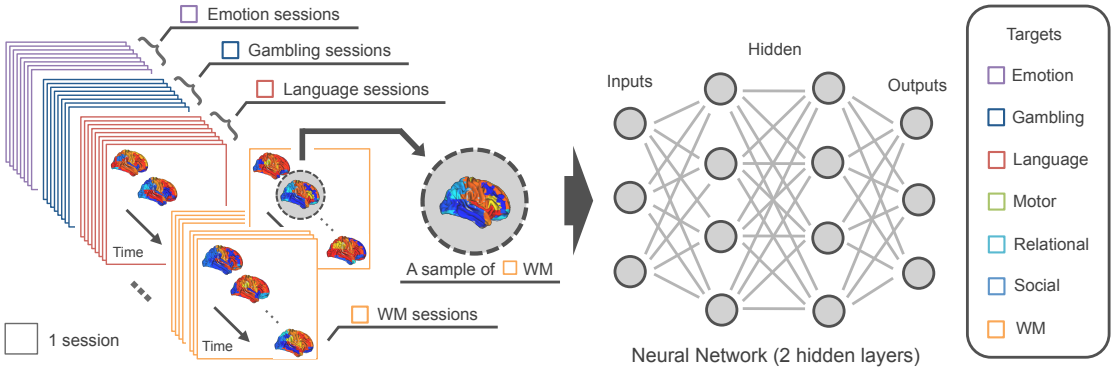

Figure 7: We trained DNNs with the input being the fMRI scans and the output being their labeled task classes.

validation dataset to determine the hyper parameters and to perform early stopping. We then tested the decoding accuracy of trained DNNs over the test dataset. We repeated this cycle 10 times, choosing different test and validation datasets in each iteration. In order to examine how the size of the dataset influences the decoding accuracy, we conducted the above experiments with different size of training dataset ($10, \ldots, 80$ and 479 subjects) without changing the test dataset and the validation dataset.

5.4 Decoding accuracy

First, we compared the decoding accuracy of the DNNs with those of other baseline methods using the dataset D (see Section 5.3). We trained three neural networks with one, two and three hidden layers, each with the output logistic regression layer. The baseline methods investigated in this study include logistic (softmax) regression, which corresponds to 0-hidden layer neural network and SVMs with linear kernel and RBF kernel (see Appendix B for the specification of these baseline methods). The mean decoding accuracy and its standard deviation in the *leave-10-subjects-out* cross validation are summarized in Table 5.4. The decoding accuracies of the DNN decoders not only exceeded the *prior* chance level (the true fraction of the largest class, 20.88%), but were also higher than those of the other baseline methods. In particular, the DNN with two hidden layers exhibited the best decoding accuracy of 50.74%, which was significantly higher than that of the RBF-kernel SVM ($p < 0.01$, one-sided Welch test). Linear methods, the logistic regression and the linear-kernel SVM, showed poor decoding accuracies that are comparable to the chance level. These results clearly show the advantage of nonlinear decoders over linear decoders in the subject-transfer setting, and suggest that the DNN may be more effective in extracting subject-independent features from big data than the other baseline methods.

Second, we investigated how the decoder's performance changes with the size of train-

Table 3: Subject-transfer decoding performance

Method	**Architecture**	**Mean accuracy [%] ± s.d.**
Logistic regression	116-7	20.81 ± 0.15
Support vector machine	Linear kernel	20.87 ± 0.01
Support vector machine	RBF kernel	47.97 ± 1.57
Neural network	116-500-7	48.94 ± 1.15
Neural network	116-500-500-7	**50.74** ± 1.25
Neural network	116-500-500-500-7	50.57 ± 1.31
Prior chance level		20.88

ing dataset. We trained the decoder with various sizes of training dataset, and plotted the decoding accuracy against the dataset size. In this set of experiments, we employed the DNN with two hidden layers ($L = 2$), which showed better decoding accuracy than the $L = 1$ and $L = 3$ versions over the dataset D. To evaluate the performance of a DNN decoder trained with a training set of M subjects, we used the following cross validation procedure. The setup of the cross validation is basically same as the one explained in Section 5.3. At each iteration of the cross validation procedure, we selected 10 subjects for the test set, 10 subjects for the validation set, and M subjects for the training set from the dataset D without any overlap. We repeated this process 10 times, selecting 10 entirely new subjects for the test set at each iteration. We conducted this set of iteration procedure for $M = 10, \ldots, 80$. In order to check the asymptotic trend of the decoding accuracy, we examined the $M = 479$ case as well, in which all of the 499 subjects registered in the S500 release were used. The results are displayed in Fig. 8. As the number of subjects in the training dataset increased from 10 to 80, the performance of the DNN decoder also increased, as expected. The performance was best at $M = 479$. We attribute this trend to the positive relationship between the size of the training dataset and the reliability of the subject-independent features captured by the DNN decoder.

5.5 PSA of the subject-transfer decoder

In order to extract the discriminative features in brain activities captured by brain decoder trained with the large scale database, we conducted principal sensitivity analysis (PSA). PSA is different from the standard sensitivity analysis in that it quantifies the *combinations* of aROIs used in the decoder's classification, whereas the standard sensitivity analysis computes the independent contribution of each aROI to the decoder's decision. The PSA is more suited for our purpose because there is a strong evidence of functional connectivity among aROIs [48, 49], and classifiers with high decoding accuracy are likely to capture coactivation patterns of aROIs. The expectation in equation (8) depends on the distribution q over the sample space. In order to best approximate the true distribution of q, we used the empirical distribution over the dataset of all subjects except the subjects used in D. Fig. 9(a) shows the first, second an third PSMs for

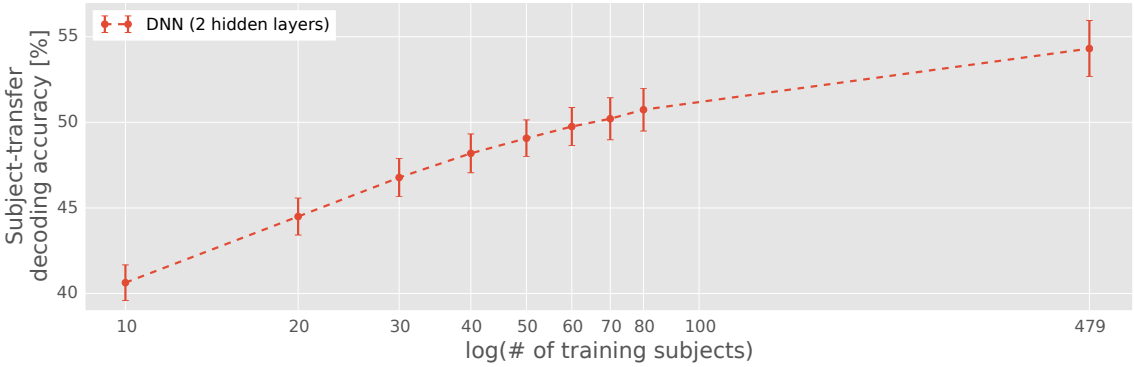

Figure 8: Mean decoding accuracy plotted against the number of training subjects (M) in log scale. Error bars indicate s.d. across ten cross validation iterations.

Emotion and Motor classes[3]. The set of PSMs presented in Fig. 9(a) is one of the 10 variants of the sets of PSMs obtained in the *leave*-10-*subjects-out* cross validation over the dataset D. These variants did not exhibit significant variation. PSMs are superior to standard sensitivity maps in that they can describe the aROIs which act oppositely in characterizing the class. Any pair of anatomical regions with different color assignments in Fig. 9(a) contributes to the classifier's decision in opposite direction. Our PSA seems to imply that the information learned by our DNN-based subject-transferable decoder has some correlation with the existing knowledge of functional connectivity supported in neuroscience. For instance, in the second PSM of Motor class and the first PSM of Social class, we can identify the two sets of functional connectivity established in previous works, namely fronto-parietal network [19] and salience network [18]. In the second PSM of Social class, we can also find a component of the default mode network [17, 16] in the left hemisphere. In addition, to quantify the similarity of PSMs, we calculated the absolute cosine similarity for each pair of the PSMs and aligned the maps by hierarchical clustering (see Fig. 9(b)). For any pair of PSMs $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, the absolute cosine similarity was calculated by $|\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle|$, where $\langle \cdot, \cdot \rangle$ is an inner product, because of $\|\boldsymbol{v}\|_2 = 1$ for each PSM $\boldsymbol{v}$. In the similarity matrix, we can confirm two large clusters, consisting mainly of first and second PSMs, respectively. This implies that the functional connectivity interpretation that we made above for the first and the second PSMs of the Social class and the the first PSM of the Motor class applies to all the other PSMs sharing the same cluster memberships (see also Appendix C). On the other hand, the sub-principal PSMs, such as the third PSMs of Emotion and Motor classes, showed task-specific features. Finally, note that many of our PSMs span large brain regions. This suggests that the subject-independent features that we extracted from the large fMRI database in our deep learning procedure are specific(common) brain-wide networks that activates during particular(all) task(s).

[3]See Appendix C for the PSMs of the other classes.

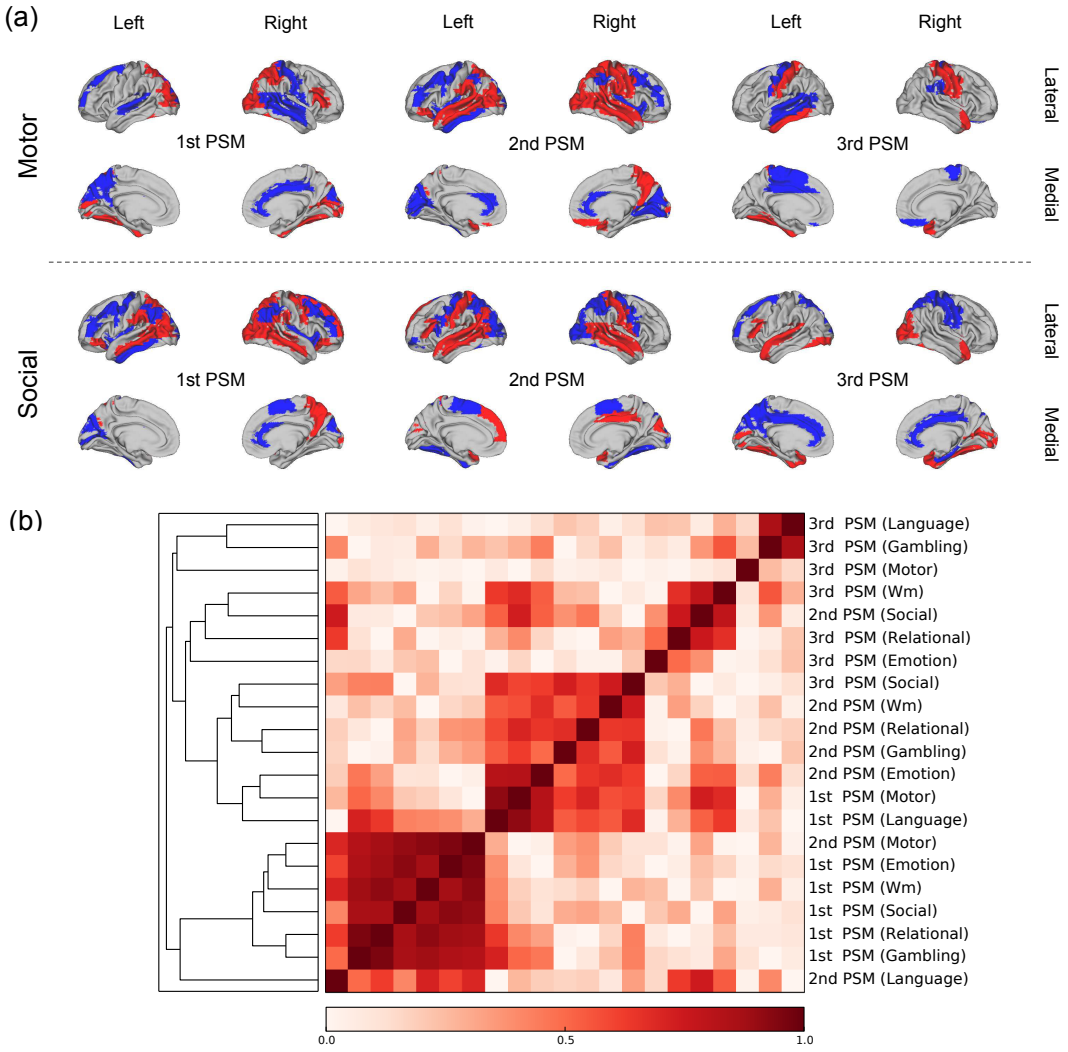

Figure 9: (a) We calculated the first, second and third PSMs for each class. The PSMs of Motor and Social classes are shown here (see Appendix C for the other classes). In the visualization of each PSM, we exclusively colored the ROIs with values that are at least one s.d. away from the mean. Red and blue indicate different signs in the PSM values, i.e., the corresponding aROIs act oppositely in characterizing the class. (b) Similarity between maps was evaluated by cosine similarity. We calculated the absolute value of cosine similarity (for definition, see the text) between all pairs of the first, second and third PSMs of the seven classes. Based on these similarity values, we applied a hierarchical clustering to the set of all the computed PSMs. The similarity matrix above is based on leaf order in the dendrogram (attached to the left side of the matrix) obtained by the hierarchical clustering. Intensity of the color indicates the degree of similarity.

6 Conclusion

We proposed a method to decompose the input space based on the sensitivity of classifiers. We assessed its performance on the classifiers trained for digit classification (Section 4) and for brain decoding as a practical application (Section 5).

In the application to digit classification, we demonstrated the PSA's three aspects of superiority to the standard sensitivity analysis [11, 12, 13]; (1) polarity of signs in a map, (2) sub-principal information, and (3) capability of sparse visualization. Furthermore, the visualization achieved with our PSA reveals at least two general aspects of the classifiers trained in the experiment. First, note in Fig. 3(b) and Fig. 6 that the first few (~ 10 out of $28 \times 28 = 784$) PSMs of the trained classifier dominate the sensitivity for the binary classification problem. Second, we see that the classifiers use these few PSMs out of 784 dimensions to solve different binary classification problems. We are thus able to see that the nonlinear classifiers in the form of neural networks solve vast number of specific classification problems (such as binary classification problems) *simultaneously and efficiently* by tuning its sensitivity to the input in a data-driven manner. One cannot attain this information with the sensitivity analysis alone.

In Section 5, we applied deep learning to a large fMRI database to classify brain activities into seven categories of human tasks. In particular, we constructed a DNN-based brain decoder aimed at classifying the brain activities from the features common to all subjects. The strength of DNN is its high ability to capture nonlinear, high dimensional features from big data. In fact, the subject-transfer decoding accuracy of our DNN was superior to those of other baseline methods, including SVMs and logistic regressions. The high performance of our DNN over the dataset acquired from a large population is indicative of the ability of our decoder to capture nonlinear discriminative features that are common to all subjects. Interestingly, when we visualized these universal discriminative features in the form of PSMs, we were able to find non-trivial associations between the features and functionally connected networks recognized in neuroscience. This observation suggests that the functional connectivity common to all subjects are playing important roles in characterizing task-specific brain activities.

With PSA, one can visualize the decomposition of the knowledge about the input space learnt by supervised classifiers. From the PSA of efficient classifiers, one might obtain a meaningful decomposition of the input space that can possibly aid us to acquire the scientific knowledge in the data which is otherwise difficult to obtain. PSA might also prove beneficial in other sciences, such as geology, atmospheric science and oceanography.

7 Acknowledgement

I would like to express my deepest gratitude to my supervisor, Professor Shin Ishii whose insightful advice and unfailing moral support were invaluable. I also wish to express my appreciation to my supervisors, Senior Lecturer Shigeyuki Oba and Assistant Professor Shinichi Maeda for their elaborated guidance as experts of machine learning. I would like to also thank Dr. Masanori Koyama, Dr. Ken Nakae and Ms. Yumi Shikauchi for their help in the preparation of this thesis and in collaboration works [50, 51, 52]. Finally, I would like to thank ATR[4] for introducing me to the field of computational neuroscience.

The data we used in Section 5 were provided by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

[4] ATR (Advanced Telecommunications Research Institute International)

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS). (2012) 1097–1105

[2] Seide, F., Li, G., Yu, D.: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In: Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech). (2011) 437–440

[3] Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014) 1701–1708

[4] Horikawa, T., Tamaki, M., Miyawaki, Y., Kamitani, Y.: Neural decoding of visual imagery during sleep. Science **340** (2013) 639–642

[5] Uberbacher, E.C., Mural, R.J.: Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. Proceedings of the National Academy of Sciences **88** (1991) 11261–11265

[6] Miyawaki, Y., Uchida, H., Yamashita, O., Sato, M., Morito, Y., Tanabe, H.C., Sadato, N., Kamitani, Y.: Visual image reconstruction from human brain activity using a combination of multiscale local image decoders. Neuron **60** (2008) 915–929

[7] LaConte, S., Strother, S., Cherkassky, V., Anderson, J., Hu, X.: Support vector machines for temporal classification of block design fMRI data. NeuroImage **26** (2005) 317–329

[8] Yamashita, O., Sato, M., Yoshioka, T., Tong, F., Kamitani, Y.: Sparse estimation automatically selects voxels relevant for the decoding of fmri activity patterns. NeuroImage **42** (2008) 1414–1429

[9] Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., Varoquaux, G.: Machine learning for neuroimaging with scikit-learn. Frontiers in Neuroinformatics **8** (2014) 14

[10] Hinton, G.E.: Dark knowledge. presented as the keynote in BayLearn (2014)

[11] Zurada, J.M., Malinowski, A., Cloete, I.: Sensitivity analysis for minimization of input data dimension for feedforward neural network. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS). Volume 6. (1994) 447–450

[12] Zurada, J.M., Malinowski, A., Usui, S.: Perturbation method for deleting redundant inputs of perceptron networks. Neurocomputing **14** (1997) 177–193

[13] Kjems, U., Hansen, L.K., Anderson, J., Frutiger, S., Muley, S., Sidtis, J., Rottenberg, D., Strother, S.C.: The quantitative evaluation of functional neuroimaging experiments: mutual information learning curves. NeuroImage **15** (2002) 772–786

[14] Rasmussen, P.M., Madsen, K.H., Lund, T.E., Hansen, L.K.: Visualization of nonlinear kernel models in neuroimaging by sensitivity maps. NeuroImage **55** (2011) 1120–1131

[15] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Muller, K.R.: How to explain individual classification decisions. The Journal of Machine Learning Research **11** (2010) 1803–1831

[16] Raichle, M.E., MacLeod, A.M., Snyder, A.Z., Powers, W.J., Gusnard, D.A., Shulman, G.L.: A default mode of brain function. Proceedings of the National Academy of Sciences **98** (2001) 676–682

[17] Raichle, M.E., Snyder, A.Z.: A default mode of brain function: a brief history of an evolving idea. NeuroImage **37** (2007) 1083–1090

[18] Taylor, K.S., Seminowicz, D.A., Davis, K.D.: Two systems of resting state connectivity between the insula and cingulate cortex. Human Brain Mapping **30** (2009) 2731–2745

[19] Cole, M.W., Reynolds, J.R., Power, J.D., Repovs, G., Anticevic, A., Braver, T.S.: Multi-task connectivity reveals flexible hubs for adaptive task control. Nature Neuroscience **16** (2013) 1348–1355

[20] Jenatton, R., Obozinski, G., Bach, F.: Structured sparse principal component analysis. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). Volume 9. (2010) 366–373

[21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. The Journal of Machine Learning Research **12** (2012) 2825–2830

[22] Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)

[23] Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). (2009) 2146–2153

[24] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012) 1–18

[25] Yukinawa, N., Oba, S., Kato, K., Ishii, S.: Optimal aggregation of binary classifiers for multiclass cancer diagnosis using gene expression profiles. IEEE/ACM Transactions on Computational Biology and Bioinformatics **6** (2009) 333–343

[26] Kontos, D., Megalooikonomou, V.: Fast and effective characterization for classification and similarity searches of 2D and 3D spatial region data. Pattern Recognition **38** (2005) 1831–1846

[27] Haxby, J.V., Gobbini, M.I., Furey, M.L., Ishai, A., Schouten, J.L., Pietrini, P.: Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science **293** (2001) 2425–2430

[28] Cox, D.D., Savoy, R.L.: Functional magnetic resonance imaging (fMRI)"brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex. NeuroImage **19** (2003) 261–270

[29] Kamitani, Y., Tong, F.: Decoding the visual and subjective contents of the human brain. Nature Neuroscience **8** (2005) 679–685

[30] Shibata, K., Watanabe, T., Sasaki, Y., Kawato, M.: Perceptual learning incepted by decoded fMRI neurofeedback without stimulus presentation. Science **334** (2011) 1413–1415

[31] LaConte, S.M.: Decoding fMRI brain states in real-time. Neuroimage **56** (2011) 440–454

[32] Sitaram, R., Veit, R., Stevens, B., Caria, A., Gerloff, C., Birbaumer, N., Hummel, F.: Acquired control of ventral premotor cortex activity by feedback training an exploratory real-time fMRI and TMS study. Neurorehabilitation and Neural Repair **26** (2012) 256–265

[33] Sitaram, R., Caria, A., Veit, R., Gaber, T., Rota, G., Kuebler, A., Birbaumer, N.: FMRI brain-computer interface: a tool for neuroscientific research and treatment. Computational Intelligence and Neuroscience **2007** (2007) 25487

[34] Lemm, S., Blankertz, B., Dickhaus, T., Müller, K.R.: Introduction to machine learning for brain imaging. Neuroimage **56** (2011) 387–399

[35] Fazli, S., Popescu, F., Danóczy, M., Blankertz, B., Müller, K.R., Grozea, C.: Subject-independent mental state classification in single trials. Neural Networks **22** (2009) 1305–1312

[36] Raizada, R.D., Connolly, A.C.: What makes different people's representations alike: neural similarity space solves the problem of across-subject fMRI decoding. Journal of Cognitive Neuroscience **24** (2012) 868–877

[37] Marquand, A.F., Brammer, M., Williams, S.C., Doyle, O.M.: Bayesian multi-task learning for decoding multi-subject neuroimaging data. NeuroImage **92** (2014) 298–311

[38] Van Essen, D.C., Smith, S.M., Barch, D.M., Behrens, T.E., Yacoub, E., Ugurbil, K.: The WU-Minn human connectome project: an overview. NeuroImage **80** (2013) 62–79

[39] Glasser, M.F., Sotiropoulos, S.N., Wilson, J.A., Coalson, T.S., Fischl, B., Andersson, J.L., Xu, J., Jbabdi, S., Webster, M., Polimeni, J.R., et al.: The minimal preprocessing pipelines for the Human Connectome Project. NeuroImage **80** (2013) 105–124

[40] Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., Mazoyer, B., Joliot, M.: Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. NeuroImage **15** (2002) 273–289

[41] Barch, D.M., Burgess, G.C., Harms, M.P., Petersen, S.E., Schlaggar, B.L., Corbetta, M., Glasser, M.F., Curtiss, S., Dixit, S., Feldt, C., et al.: Function in the human connectome: Task-fMRI and individual differences in behavior. NeuroImage **80** (2013) 169–189

[42] Hariri, A.R., Tessitore, A., Mattay, V.S., Fera, F., Weinberger, D.R.: The amygdala response to emotional stimuli: a comparison of faces and scenes. NeuroImage **17** (2002) 317–323

[43] Delgado, M.R., Nystrom, L.E., Fissell, C., Noll, D., Fiez, J.A.: Tracking the hemodynamic responses to reward and punishment in the striatum. Journal of Neurophysiology **84** (2000) 3072–3077

[44] Binder, J.R., Gross, W.L., Allendorfer, J.B., Bonilha, L., Chapin, J., Edwards, J.C., Grabowski, T.J., Langfitt, J.T., Loring, D.W., Lowe, M.J., et al.: Mapping anterior temporal lobe language areas with fMRI: a multicenter normative study. NeuroImage **54** (2011) 1465–1475

[45] Buckner, R.L., Krienen, F.M., Castellanos, A., Diaz, J.C., Yeo, B.T.: The organization of the human cerebellum estimated by intrinsic functional connectivity. Journal of Neurophysiology **106** (2011) 2322–2345

[46] Castelli, F., Happé, F., Frith, U., Frith, C.: Movement and mind: a functional imaging study of perception and interpretation of complex intentional movement patterns. NeuroImage **12** (2000) 314–325

[47] Wheatley, T., Milleville, S.C., Martin, A.: Understanding animate agents distinct roles for the social network and mirror system. Psychological Science **18** (2007) 469–474

[48] Buckner, R.L., Krienen, F.M., Yeo, B.T.: Opportunities and limitations of intrinsic functional connectivity MRI. Nature Neuroscience **16** (2013) 832–837

[49] Cole, M.W., Bassett, D.S., Power, J.D., Braver, T.S., Petersen, S.E.: Intrinsic and task-evoked network architectures of the human brain. Neuron **83** (2014) 238–51

[50] Koyamada, S., Shikauchi, Y., Nakae, K., Ishii, S.: Construction of subject-independent brain becoders for human fMRI with deep learning. In: Proceedings of the International Conference on Data Mining, Internet Computing, and Big Data (BigData2014). (2014) 60–68

[51] Koyamada, S., Koyama, M., Nakae, K., Ishii, S.: Principal sensitivity analysis. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). (to appear)

[52] Koyamada, S., Shikauchi, Y., Nakae, K., Koyama, M., Ishii, S.: Sensitivity analysis of neural network based subject-transfer decoder on fMRI big data. arXiv preprint arXiv:1502.00093 (submitted)

[53] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research **12** (2011) 2825–2830

Appendix A: All PSMs of digit recognition

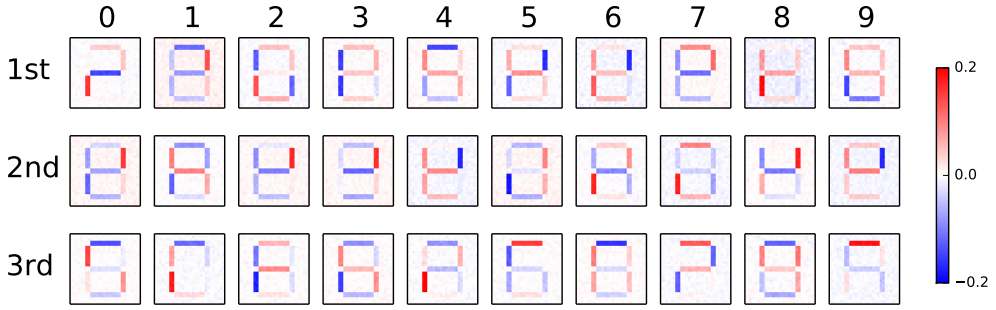

Figure 10: First, second and third PSMs of the classifier trained on the artificial dataset.

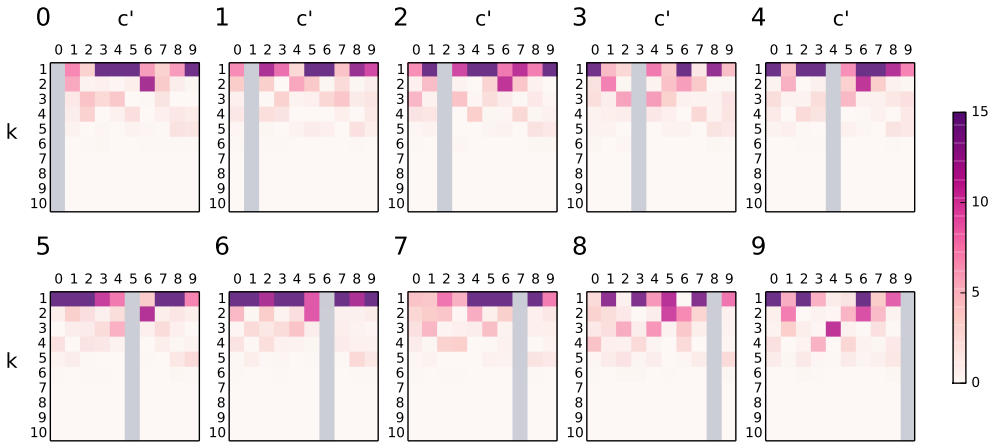

Figure 11: $s_{c,c'}(\boldsymbol{v}_k)$ on the artificial dataset.

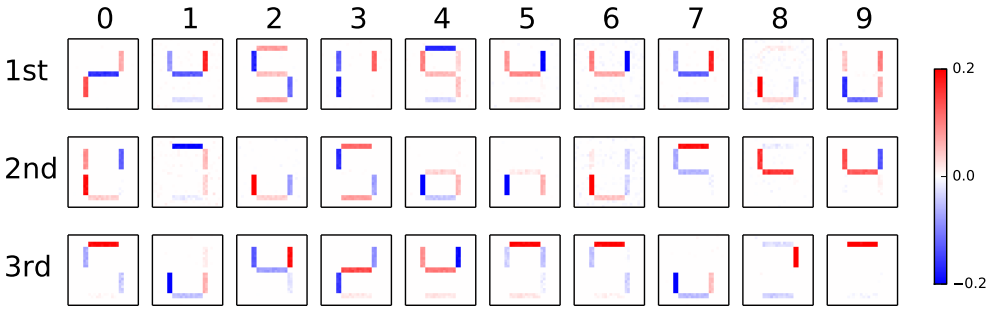

Figure 12: Results of the sparse PSA on the classifiers trained on the artificial dataset.

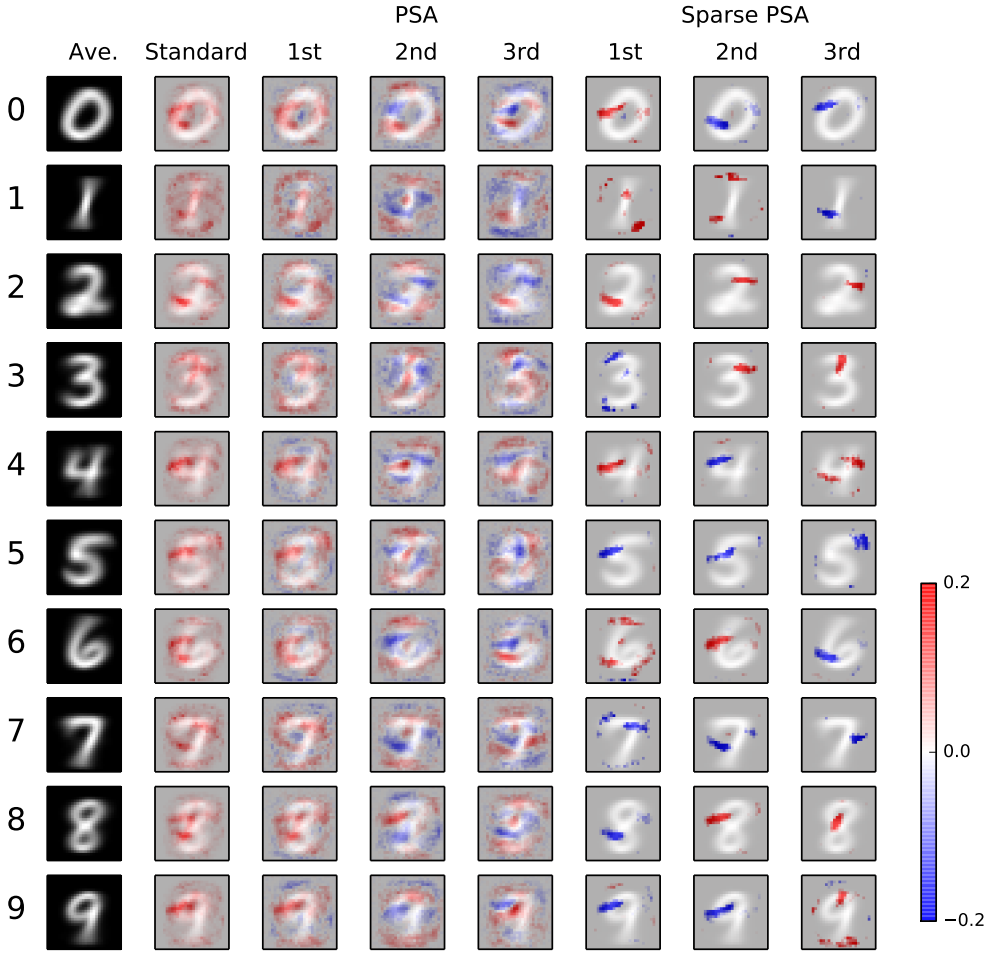

Figure 13: Average, standard sensitivity maps, PSMs, and sparse PSMs on MNIST data.

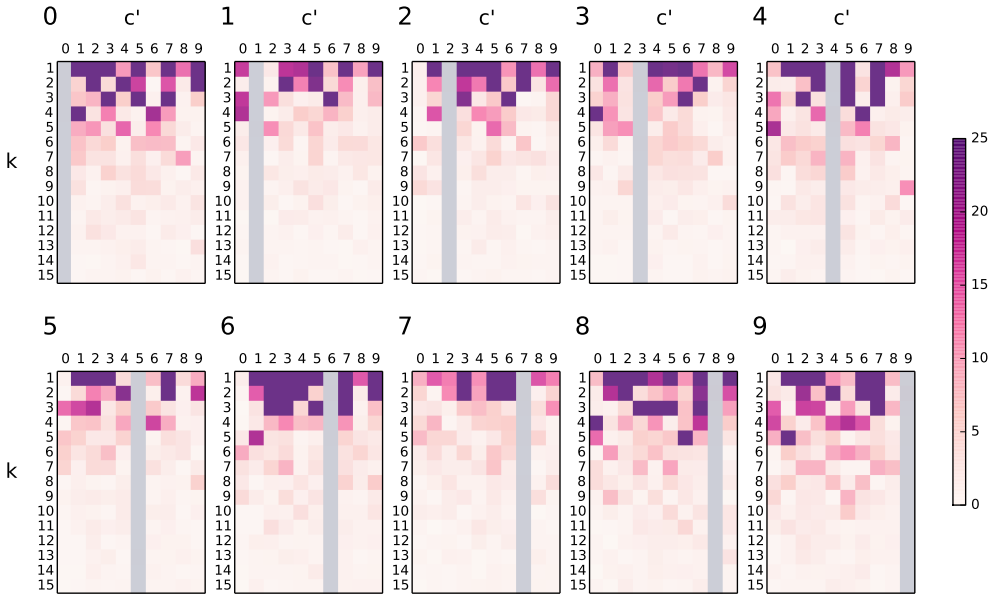

Figure 14: $s_{c,c'}(\boldsymbol{v}_k)$ on MNSIT dataset.

Appendix B: Specifications of baseline methods

Each version of SVM consists of seven one-versus-the-rest classifiers. As for the SVMs, we used scikit-learn [53]. Hyper-parameters were chosen to maximize the decoding accuracy over the validation dataset (see Section 5.3); we heuristically prepared nine sets of hyper-parameters for each method, and adopted the one that achieved the best decoding accuracy for the validation dataset. The hyper-parameter for the logistic regression was the learning rate in the MSGD. The hyper-parameter for the linear-kernel SVM was the regularization strength C used in the scikit-learn. The RBM-kernel SVM was dependent on a pair of hyper-parameters, the regularization strength C and the kernel width γ. We considered 3 values each for C and γ, and examined all nine pairs for the best choice.

Appendix C: All PSMs of subject-transfer decoder

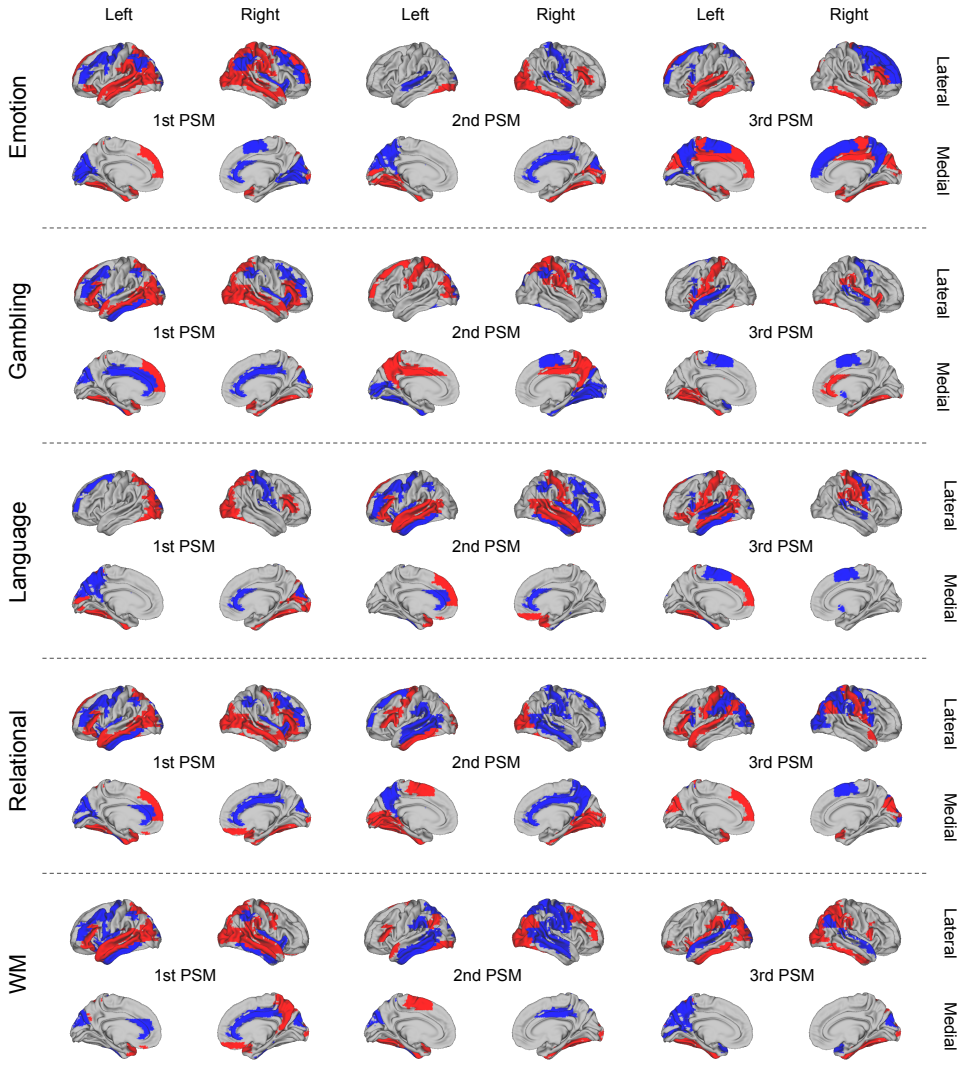

Figure 15: PSMs of the other classes (Emotion, Gambling, Language, Relational and WM).