Principal Sensitivity Analysis

Sotetsu Koyamada[12] Masanori Koyama[1] Ken Nakae[1] and Shin Ishii[12]

[1] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[2] ATR Cognitive Mechanisms Laboratories, Kyoto, Japan
`koyamada-s@sys.i.kyoto-u.ac.jp, ishii@i.kyoto-u.ac.jp`

Abstract. We present a novel algorithm (Principal Sensitivity Analysis; PSA) to analyze the knowledge of the classifier obtained from supervised machine learning techniques. In particular, we define principal sensitivity map (PSM) as the direction on the input space to which the trained classifier is most sensitive, and use analogously defined k-th PSM to define a basis for the input space. We train neural networks with artificial data and real data, and apply the algorithm to the obtained supervised classifiers. We then visualize the PSMs to demonstrate the PSA's ability to decompose the knowledge acquired by the trained classifiers.

1 Introduction

Machine learning is a powerful methodology to construct efficient and robust predictors and classifiers. Literature suggests its ability in the supervised context not only to reproduce "intuition and experience" based on human supervision [1], but also to successfully classify the objects that humans cannot sufficiently classify with inspection alone [2,3]. This work is motivated by the cases in which the machine classifier eclipses the human decisions. We may say that this is the case in which the classifier holds more knowledge about the 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. The superiority of nonlinear machine learning techniques 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 classifiers [4]. One might therefore be motivated to enhance understanding of subject classes by studying the way the trained machine acquires the information.

Unfortunately, trained classifiers are often so complex that they defy human interpretation. Although some efforts have been made to "visualize" the classifiers [5,6], there is still much room left for improvement. The machine learning techniques in neuroimaging, for example, prefer linear kernels to nonlinear kernels because of the lack of visualization techniques [7]. For the visualization

of high-dimensional feature space of machine learners, Zurada et al. [8,9] and Kjems et al. [10] presented seminal works. Zurada et al. developed "sensitivity analysis" in order to "delete unimportant data components for feedforward neural networks." Kjems et al. visualized Zurada's idea as "sensitivity map" in the context of neuroimaging. 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).

2 Methods

2.1 Conventional sensitivity analysis

Before introducing the PSA, we describe the original sensitivity map introduced in [10]. 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

$$s_i := \int \left(\frac{\partial f(\boldsymbol{x})}{\partial x_i} \right)^2 q(\boldsymbol{x}) d\boldsymbol{x}, \qquad (1)$$

where q is the distribution over the input space. In actual implementation, the integral (1) is computed with the empirical distribution q of the test dataset. Now, the vector

$$\boldsymbol{s} := (s_1, \ldots, s_d) \qquad (2)$$

of these values will give us an intuitive measure for the degree of importance that the classifier attaches to each input. Kjems et al. [10] defined $\boldsymbol{s}$ as **sensitivity map** over the set of input features.

2.2 Sensitivity in arbitrary direction

Here, we generalize the definition (1). 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 $\mathbb{R}^d$:

$$s(\boldsymbol{v}) := \int \left(\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{v}} \right)^2 q(\boldsymbol{x}) \, d\boldsymbol{x}. \tag{3}$$

Recall that the directional derivative is defined by

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{v}} := \sum_{i=1}^d v_i \frac{\partial f(\boldsymbol{x})}{\partial x_i}.$$

Note that when we define the *sensitivity inner product*

$$\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle_s := \int \left(\frac{\partial f(\boldsymbol{x})}{\partial x_i} \right) \left(\frac{\partial f(\boldsymbol{x})}{\partial x_j} \right) q(\boldsymbol{x}) \, d\boldsymbol{x}, \tag{4}$$

we can rewrite $s(\boldsymbol{v})$ with the corresponding *sensitivity norm*, as follows:

$$\begin{aligned} \|\boldsymbol{v}\|_s^2 &:= \langle \boldsymbol{v}, \boldsymbol{v} \rangle_s \\ &= \left\langle \sum_i v_i \boldsymbol{e}_i, \sum_j v_j \boldsymbol{e}_j \right\rangle_s \\ &= \sum_{i,j} v_i v_j \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle_s . \end{aligned} \tag{5}$$

This inner product defines the kernel metric corresponding to the positive definite matrix $\boldsymbol{K}$ with ij-th entry given by $K_{ij} := \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle_s$. This allows us to write

$$s(\boldsymbol{v}) = \boldsymbol{v}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{v}. \tag{6}$$

2.3 Principal sensitivity map and PSA

The classical setting (2) was developed in order to quantify the sensitivity of f with respect to each individual input feature. 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 $\boldsymbol{v}$:

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{v}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{v} \\ \text{subject to} \quad & \boldsymbol{v}^{\mathrm{T}} \boldsymbol{v} = 1. \end{aligned} \tag{7}$$

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 (2) 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 3.1 we will demonstrate the benefit of this extra information.

Principal Sensitivity Analysis (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 $\boldsymbol{V}$:

$$\begin{aligned} \text{maximize} \quad & \text{Tr}\left(\boldsymbol{V}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{V}\right) \\ \text{subject to} \quad & \boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_j = \delta_{ij}, \end{aligned} \tag{8}$$

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 (8) is given by the covariance $E\left[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right]$, where $\boldsymbol{x}$ is the centered random variable. Note that in our particular case, if we put

$$\boldsymbol{r}(\boldsymbol{x}) := \left(\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}\right), \ldots, \left(\frac{\partial f(\boldsymbol{x})}{\partial x_d}\right)\right)^{\mathrm{T}}, \tag{9}$$

then we may write $\boldsymbol{K} = \int \boldsymbol{r}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})^{\mathrm{T}}q(\boldsymbol{x})\,d\boldsymbol{x} = E\left[\boldsymbol{r}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})^{\mathrm{T}}\right]$. We see that our algorithm can thus be seen as the PCA applied to the covariance of $\boldsymbol{r}(\boldsymbol{x})$ without centering.

Sparse PSA One may use the new definition (8) 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 [11] 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 $\boldsymbol{\alpha}_i$:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_i^N \|\boldsymbol{r}\left(\boldsymbol{x}_i\right) - \boldsymbol{V}\boldsymbol{\alpha}_i\|_2^2 + \lambda \sum_k^p \|\boldsymbol{v}_k\|_1 \\ \text{subject to} \quad & \|\boldsymbol{\alpha}_i\|_2 = 1, \end{aligned} \tag{10}$$

where p is the number of sensitivity maps and N is the number of samples. For the implementation, we used scikit-learn [12].

2.4 Experiments

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. 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 (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. Using the

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

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 stochastic gradient descent with constant learning rate. We also adopted a dropout method [13] 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:

$$f_c(\boldsymbol{x}) := \log P(Y = c \,|\, \boldsymbol{x}), \tag{11}$$

where Y is, in the model governing the neural network, a random variable representing the class that the classifier assigns to the input $\boldsymbol{x}$. We then constructed the PSM 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

3   Results

3.1   PSA of 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 PSM and standard sensitivity map, 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 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 (1st) 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.

Fig. 2. (a) The standard sensitivity maps. (b) The PSMs.

Next, we will show the benefits of sub-principal sensitivity maps computed from PSA. Fig. 3(a) shows the 1st PSM through the 3rd PSM for the numerals 0 and 9.[3] 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:

$$s_{c,c'}(\boldsymbol{v}) = \int \left(\frac{\partial f_c(\boldsymbol{x})}{\partial \boldsymbol{v}} \right)^2 q_{c,c'}(\boldsymbol{x}) d\boldsymbol{x}, \tag{12}$$

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 map, note that this value for

---

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

$(c, c') = (9, 4)$ can also be pictorially written as

$$\lim_{\varepsilon \to 0} E_{\{9,4\}} \left[\left(\frac{\log P\left(Y = \boxed{9} \mid \boxed{\blacksquare} + \varepsilon \boldsymbol{v}\right) - \log P\left(Y = \boxed{9} \mid \boxed{\blacksquare}\right)}{\varepsilon} \right)^2 \right], \quad (13)$$

where $\boldsymbol{v}$ can be the 3rd PSM of class 9, $\boxed{5}$, 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 $k \in \{1, \dots, 10\}$. 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 3rd PSM is more helpful than the 1st PSM for distinguishing 4 from 9. We can actually verify this fact by observing that the 3rd 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 1st 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.

Fig. 3. (a) 1st $\sim$ 3rd PSMs of the classifier outputs f_c for the numerals 0 and 9. (b) $s_{c,c'}(\boldsymbol{v}_k)$ for $c \in \{0, 9\}$, $k \in \{1, \dots, 10\}$, and $c' \in \{0, \dots, 9\}\backslash\{c\}$.

Finally, we demonstrate the usefulness of formulation (8) in the construction of sparse and intuitive sensitivity map. Fig. 4 depicts the sensitivity maps obtained from the application of our sparse PSA in (10) 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.

Fig. 4. Results of the sparse PSA on the classifiers f_c with $p = 3$ 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.

3.2 PSA of 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 displacement 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 neuroimaing, 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 [14]. Likewise, in meteorology, 3D dataset is often translated into voxel structures, and a group of voxels may represent geographical region of specific terrain [15]. 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.

Fig. 5. Standard sensitivity map, PSA, and sparse PSA 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.

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

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

4 Discussion

We proposed a method to decompose the input space based on the sensitivity of classifiers. We assessed its performance on classifiers trained with artificial data and MNIST data. The visualization achieved with our PSA reveals at least two general aspects of the classifiers trained in this experiment. First, note in Fig. 3(b) and Fig. 6 that the first few (~ 10) PSMs of the trained classifier dominate the sensitivity for the binary classification problem. Second, we see that the classifier use these few PSMs out of 784 dimensions to solve different binary classification problems. We are thus able to see that the nonlinear classifiers of the neural network 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 standard sensitivity map [8,9,10] alone. With PSA, one can visualize the

decomposition of the knowledge about the input space learnt by the classifier. From the PSA of efficient classifier, one may obtain a meaningful decomposition of the input space that can possibly aid us in solving wide variety of problems. In medical science, for example, PSA might identify a combination of the biological regions that are helpful in diagnosis. PSA might also prove beneficial in sciences using voxel based approaches, such as geology, atmospheric science, and oceanography.

We may incorporate the principle of the PSA into existing standard statistical methods. A group Lasso analogue of the PSA, which is currently under our development, may enhance the interpretability of the visualization even further by identifying sets of voxels with biological organs, geographical location, etc. By improving its interpretability, PSA and the PSA-like techniques might significantly increase the applicability of machine learning techniques to various high-dimensional problems.

References

1. 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
2. Horikawa, T., Tamaki, M., Miyawaki, Y., Kamitani, Y.: Neural decoding of visual imagery during sleep. Science **340** (2013) 639–642
3. 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
4. Hinton, G.E.: Dark knowledge. Presented as the keynote in BayLearn (2014)
5. 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
6. 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
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. 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
9. Zurada, J.M., Malinowski, A., Usui, S.: Perturbation method for deleting redundant inputs of perceptron networks. Neurocomputing **14** (1997) 177–193
10. 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
11. 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

12. 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

13. 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

14. 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

15. 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

Appendix

In this section we list the figures that we omitted in the main text.

Fig. 7. 1st $\sim$ 3rd PSMs of the classifier trained on the artificial dataset.

Fig. 8. $s_{c,c'}(\boldsymbol{v}_k)$ on the artificial dataset.

Fig. 9. Results of the sparse PSA on the classifiers trained on the artificial dataset.

Fig. 10. Average, standard sensitivity map, PSA, and sparse PSA on MNIST data.

Fig. 11. $s_{c,c'}(\boldsymbol{v}_k)$ on MNSIT dataset.