JMLR: Workshop and Conference Proceedings 44 (2015) 130-144 NIPS 2015

The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"

FEAST at Play: Feature ExtrAction using Score function Tensors

Majid Janzamin[*] MJANZAMI@UCI.EDU
Hanie Sedghi[*] SEDGHIH@UCI.EDU
U.N. Niranjan[*] UN.NIRANJAN@UCI.EDU
Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Department of Electrical Engineering and Computer Science
University of California, Irvine
Engineering Hall, Room 4406
Irvine, CA 92697, USA

Editor: Dmitry Storcheus

Abstract

Feature learning forms the cornerstone for tackling challenging learning problems in domains such as speech, computer vision and natural language processing. In this paper, we build upon a novel framework called FEAST(Feature ExtrAction using Score function Tensors) which incorporates generative models for discriminative learning. FEAST considers a novel class of matrix and tensor-valued feature transform, which can be pre-trained using unlabeled samples. It uses an efficient algorithm for extracting discriminative information, given these pre-trained features and labeled samples for any related task. The class of features it adopts are based on higher-order score functions, which capture local variations in the probability density function of the input. We employ efficient spectral decomposition algorithms (on matrices and tensors) for extracting discriminative components. The advantage of employing tensor-valued features is that we can extract richer discriminative information in the form of overcomplete representations (where number of discriminative features is greater than input dimension). In this paper, we provide preliminary experiment results on real datasets.

Keywords: Discriminative features, spectral decomposition, tensor representations, score function

1. Introduction

Having good features or representations of the input data is critical to achieving good performance in challenging machine learning tasks in domains such as speech, computer vision and natural language processing. Traditionally, feature engineering relied on carefully hand-crafted features, tailored towards a specific task, which is laborious and time-consuming. Instead, the modern approach is to automatically learn good feature representations from data. An overarching principle behind automated feature learning is the discovery of a latent representations that can succinctly summarize the observed data. Examples include sparse coding (Raina et al., 2007), independent component analysis (ICA) (Le et al., 2011),

. [*] These authors contributed equally to this work.

Fisher kernels (Jaakkola et al., 1999) and auto-encoders (Bengio et al., 2013). Many of these approaches are unsupervised, and can therefore exploit the vast amounts of unlabeled samples present in many domains. The idea is to then use these learnt representations in classification tasks.

It is often the case that the learnt representations through unsupervised learning may not be relevant for the specific classification task, since they are not trained using label information. Moreover, learning complex latent representations of the input is challenging and error-prone, and directly using it in a classifier may degrade performance (Cozman et al., 2002). Are there principled approaches to first extract discriminative features from pre-trained representations, before employing them in a classifier?

It has been postulated that humans mostly learn in an unsupervised manner (Raina et al., 2007), gathering “common-sense” or “general-purpose” knowledge, without worrying about any specific goals. Indeed, when faced with a specific task, humans can quickly and easily extract relevant information from the accrued general-purpose knowledge. Can we design machines with similar capabilities? Are there a class of pre-trained representations which serve as “general-purpose” knowledge? Are there efficient algorithms to quickly specialize them for a task at hand? Are there theoretical guarantees for such methods?

Recently, Janzamin et al. (2014) have provided positive answers to all these questions. They have provided a method for extracting discriminative features in a semi-supervised manner. They proposed a new method called FEAST (Feature ExtrAction using Score function Tensors) which adopts a class of matrix and tensor-valued “general-purpose” features, which can be pre-trained using unlabeled samples. When presented with labeled samples, they leverage these pre-trained features to extract discriminative information using efficient spectral decomposition algorithms.

They incorporate a generative model for the input and consider a class of features based on *score functions* of the input. Score functions are normalized derivatives of the input probability density function (pdf) and capture its “local manifold structure” (Janzamin et al., 2014). While the first-order score function (i.e., first order derivative of the pdf) is a vector, assuming a vector input, the higher-order score functions are matrices and tensors. These score functions can be pre-trained in an unsupervised manner, without the need for labels (Janzamin et al., 2014). This is very useful as in practice we have access to more unlabeled data than labeled data.

1.1 Summary of results

In this paper, we adopt a new method called FEAST (Feature ExtrAction using Score function Tensors) for extracting discriminative features proposed by Janzamin et al. (2014). As explained above, they propose score functions which can be pre-trained in an unsupervised manner, without the need for labels. In this work, we demonstrate how these score functions can be efficiently approximated using denoising auto-encoders (DAE), without assuming any parametric class of models. In addition, we provide preliminary experiments on real datasets. We describe our summary of results by first providing a motivating example on how these new features can be used for a classification task.

Motivating Toy Example with Swiss Roll: In Figure 1, we plot a two dimen- sional Swiss roll, which is a simple non-linear manifold. Reconstructing it with a denoising auto-encoder (DAE) (in an unsupervised manner) results in significant distortions. However, DAE provides us with more information. We show that it allows for reconstructing the score function. The first order score function is a vector field and the second order score function is a matrix field. Therefore, we use DAE for approximating vector and matrix fields. We plot the first and second order score functions in Figures 1(a),(b). The arrows represent the vector and matrix fields.

Given these pre-trained score-function features and labeled samples, Janzamin et al. (2014) extract discriminative information based on the *spectral methods*. Spectral methods involve various iterative operations to yield matrix and tensor features. They are fast, embarrassingly parallel and can handle huge datasets with billions of dimensions. They employ spectral methods to decompose moment matrices/tensors which encode the correlation between labels and input score functions. The extracted eigenvectors from label-score function moments are then used to train the classifiers as follows: we project the input data along the extracted eigenvectors (and apply suitable non-linearity) and then train a standard classifier such as a linear/non-linear SVM. Using such simple classifiers enables fast inference at test time. The end-to-end framework (Janzamin et al., 2014) is provided in Figure 2 and it is referred as FEAST (Feature ExtrAction using Score function Tensors). Note that since the extracted eigenvectors bear discriminative directions, throughout the paper we refer to them as discriminative features.

Spectral methods have previously been mostly used on *raw* moments of input, such as pairwise correlations, skewness, kurtosis, and so on, as in the case of principal component analysis (PCA), canonical correlation analysis (CCA), and tensor decompositions (Anandkumar et al., 2014a). In (Janzamin et al., 2014), they instead, transform the input into score functions and then correlate them with the labels. These score function transformations are in general non-linear and therefore, enable us to capture higher order moments of data, which are argued to be crucial for classification (Bruna and Mallat, 2013). Instead of using a fixed non-linear transform, as done in standard frameworks such as kernels (Scholkopf and Smola, 2001), wavelets (Bruna and Mallat, 2013), Janzamin et al. (2014) learn the transform from data itself and capture local variations of the input through score functions.

Why extract discriminative features in this manner? What is the nature of information extracted? Janzamin et al. (2014) establish that the moments encoding the correlation between the label and input score functions yield information about how the label varies as a function of the input. Concretely, they prove that the label-score function moment is

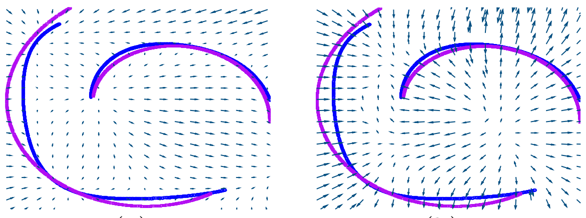
(a) (b)

Figure 1: Reconstruction of Swiss roll using denoising auto-encoder (DAE) and extracting discriminative directions using FEAST method (Janzamin et al., 2014). In (a), (b), we plot score functions $\mathcal{S}_m(x)$ approximated using DAE. (a) Vector field $\mathcal{S}_1(x)$. (b) Top eigenvector of matrix field $\mathcal{S}_2(x)$.

equal to the (expected) derivative of the label, as a function of the input. By considering the m^{th} order score function, they obtain the m^{th} order derivative of the label function. Thus, these label-score function moments capture variations of the label function, and are therefore informative for discriminative tasks.

Why tensor decompositions are needed? While the first order score function is a vector, higher order score functions are matrices and tensors. By going to higher order scores, we can extract more information about the label function variation by decomposing the label-score function moments. The use of tensor score functions allows us to extract overcomplete representations since the rank of the tensor can exceed its dimensions. Note that this is not the case with matrices. It has been argued that having overcomplete representations is crucial to getting good classification performance (Coates et al., 2011). This is verified in our experiments. For details, see Section 4. Another advantage of the tensor methods is that they do not suffer from spurious local optima, compared to typical non-convex problems such as expectation maximization or backpropagation. Anandkumar et al. (2014a,b,c) provide detailed analysis of efficient tensor decomposition algorithms. Thus, we can leverage the latest advances in spectral methods for efficient extraction of discriminative information from moment tensors. At the same time, going to tensors does not result in significant increase in computational complexity. In fact, FEAST has computational complexity which is logarithmic in number of relevant dimension when using a polynomial number of processors.

Experiment Results: In this paper, we use denoising auto-encoders for estimating the score function. We first illustrate score functions of different orders for Swiss roll dataset and also notice that although the output of DAE is noisy, if we think of each piece of Swiss roll as a class, FEAST can correctly classify the two classes when we use the third order score function. Next, we perform FEAST on MNIST and obtain competitive results with a simple linear SVM without additional tuning of parameters. In addition, we use Gaussian kernel SVM and obtain better results compared to linear SVM. In both cases, we obtain better performance in the overcomplete setting and this is only possible when using higher order score functions and tensor moments. Moreover, by increasing the rank of the tensor we achieve better classification performance. This is the case as the rank of the tensor is equal to the number of discriminative features and having more discriminative features improves the classification accuracy.

2. Related Work

Feature learning forms a cornerstone of modern machine learning (Bengio et al., 2013). Appropriate feature transformations are crucial for obtaining good classification performance. While deep learning is presented as an efficient framework for feature extraction, a theoretical understanding is mostly lacking. In contrast, in this paper, we use FEAST which is a feature learning framework that has theoretical backing. FEAST employs a generative model of the input, which is also done in Fisher kernels (Jaakkola et al., 1999). However, in practice, a common complaint is that these generative features are not discriminative for the task at hand. Previous solutions have prescribed joint training discriminative features using labeled samples, in conjunction with unlabeled samples (Mairal et al., 2009; Maaten, 2011; Wang et al., 2013). However, the resulting optimization problems are complex and expensive to run, may not converge to good solutions, and have to be re-trained for each new task. FEAST is an alternative approach to extract discriminative features from generative models by using efficient spectral decomposition algorithms on moment matrices and tensors.

We now contrast FEAST with previous moment-based approaches for discriminative learning, which consider moments between the label and raw input, e.g., canonical correlation analysis (CCA) and generalized eigen analysis (Karampatziakis and Mineiro, 2014). In contrast, FEAST constructs cross-moments between the label and the score function features. We show that using score function features is crucial to mining discriminative information with provable guarantees. In addition, Karampatziakis and Mineiro (2014) obtain discriminative features by only incorporating pairwise information between different label classes. Their method is not scalable to large output spaces since it is quadratic in the number of classes. In contrast, by using tensors, we can incorporate all label dimensions together, and obtain rich discriminative features.

2.1 Notation

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Throughout this paper, $\nabla_x^{(m)}$ denotes the m-th order derivative w.r.t. variable x and notation $\otimes$ represents tensor (outer) product. A real *r-th order tensor* $T \in \bigotimes_{i=1}^{r} \mathbb{R}^{d_i}$ is a member of the outer product of Euclidean spaces $\mathbb{R}^{d_i}$, $i \in [r]$. The different dimensions of the tensor are referred to as *modes*. For convenience, we restrict to the case where $d_1 = d_2 = \cdots = d_r = d$, and simply write $T \in \bigotimes^r \mathbb{R}^d$. As is the case for vectors (where $r = 1$) and matrices (where $r = 2$), we may identify a r-th order tensor with the r-way array of real numbers $[T_{i_1,i_2,\ldots,i_r} : i_1, i_2, \ldots, i_r \in [d]]$, where $T_{i_1,i_2,\ldots,i_r}$ is the $(i_1, i_2, \ldots, i_r)$-th coordinate of T with respect to a canonical basis. Also, the notation $u^{\otimes 3}$ is the short form for $u \otimes u \otimes u$.

3. Overview of the Method

In this section, we first review the FEAST framework (Janzamin et al., 2014) for Feature ExtrAction using Score function Tensors, then we elaborate on our approach to implementing FEAST. The end-to-end framework is presented in Figure 2. Figure 3 shows a proof of concept of performing FEAST on MNIST data set[1]. In a snapshot, FEAST method consists of two parts, first we learn some function of the input called *score function* in an unsupervised manner. Next, using labeled data we form the cross-moment between label and score function of the input. We then run tensor decomposition on the calculated moment which provides us with discriminative features.

Score Function $\mathcal{S}_m(x)$: Let $p(x)$ denote the joint pdf of random vector $x \in \mathbb{R}^d$. Janzamin et al. (2014) denote $\mathcal{S}_m(x)$ as the m-th order score function, which is given by

$$\mathcal{S}_m(x) := (-1)^m \frac{\nabla_x^{(m)} p(x)}{p(x)}, \tag{1}$$

1. `http://yann.lecun.com/exdb/mnist/`

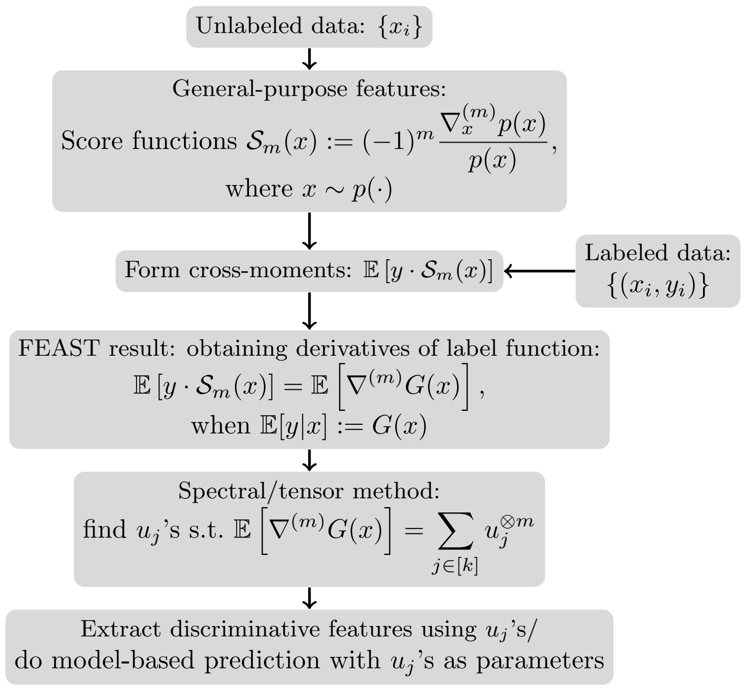
Unlabeled data: $\{x_i\}$
General-purpose features:
Score functions $\mathcal{S}_m(x) := (-1)^m \frac{\nabla_x^{(m)} p(x)}{p(x)}$, where $x \sim p(\cdot)$
Form cross-moments: $\mathbb{E}[y \cdot \mathcal{S}_m(x)]$
Labeled data: $\{(x_i, y_i)\}$
FEAST result: obtaining derivatives of label function: $\mathbb{E}[y \cdot \mathcal{S}_m(x)] = \mathbb{E}\left[\nabla^{(m)} G(x)\right]$, when $\mathbb{E}[y|x] := G(x)$
Spectral/tensor method: find u_j's s.t. $\mathbb{E}\left[\nabla^{(m)} G(x)\right] = \sum_{j \in [k]} u_j^{\otimes m}$
Extract discriminative features using u_j's/ do model-based prediction with u_j's as parameters

Figure 2: Overview of the FEAST framework (Janzamin et al., 2014)

where $\nabla_x^{(m)}$ denotes the m-th order derivative operator w.r.t. variable x. The estimation of score functions $\mathcal{S}_m(x)$ is performed in an unsupervised manner using unlabeled samples $\{x_i\}$.

Score function estimation with auto-encoders: One framework for estimating score functions is via auto-encoders. We employ them in our experiments due to their simplicity. In order to estimate the score function of the input we either need to assume a generative model for input features or use a model-free method such as auto-encoder. Both methods are applicable in FEAST. In this paper, we elaborate on the latter. Recall that auto-encoders attempt to find encoding and decoding functions which minimize the reconstruction error under noise (the so-called denoising auto-encoders or DAE). In an auto-encoder, let $f(x)$ be the encoding function and $g(h)$ be the decoding function that maps $f(x)$ back to the input space. The reconstruction function $r(x) = g(f(x))$ is the composition of encoder and decoder functions. Denoising is a form of regularization and the goal of DAE is to penalize the error made in $r(x)$ as a prediction of x denoted by $\mathcal{L}_{\text{DAE}} = \mathbb{E}[\|x - r(x+\epsilon)\|^2]$, where ϵ is the Gaussian corruption noise source. This is an unsupervised framework involving only unlabeled samples. Alain and Bengio (2012) argue that the DAE approximately learns the score function of the input, as the noise variance goes to zero.

$$r_\sigma^*(x) = x + \sigma^2 \nabla_x \log p(x) + o(\sigma^2) \quad \text{as} \quad \sigma \to 0, \tag{2}$$

where $r_\sigma^*(x)$ is the optimal reconstruction function and σ is noise variance. We extend this framework to estimate higher order score functions. In Section 3.1, we establish that DAEs can correctly estimate the higher order score functions in the limit as the noise goes to zero.

After estimating the score function, we form the cross moment $\mathbb{E}[y \otimes \mathcal{S}_m(x)]$ between label y and (higher order) score function $\mathcal{S}_m(x)$ using labeled data.

Tensor Decomposition Method: Janzamin et al. (2014) have shown that the cross-moment $\mathbb{E}\left[y \otimes \mathcal{S}_m(x)\right]$ yields information of derivative of the label w.r.t. the input (which are matrices or tensors). They then find efficient representations using spectral/tensor decomposition methods. When we employ the first order score function, we carry out spectral decomposition on the cross-moment matrix (assuming y is vector). With higher order score function ($m \geq 2$), we carry out tensor decomposition, as depicted in Figure 7 in the Appendix B.3. The advantage of using tensors is that the number of the components we obtain can be more than the input dimension(this is called overcomplete setting), which is not the case with matrices. In our experiments in Section 4, we find that overcomplete representations yield better classification performance than undercomplete representations.

Classification: The output of the spectral method provides us with discriminative features $\{u_i\}$. We then form a nonlinear function of $\langle u_i, x\rangle$ and feed it to a linear SVM classifier. We can apply different non-linearities on $\langle u_i, x\rangle$. Karampatziakis and Mineiro (2014) discuss different nonlinear functions of the form $\max(0, \delta\langle u_i, x\rangle)^{\alpha/2})$. In our experiment we use the same nonlinear function with $\alpha = 2$, and notice that the performance is fairly invariant to the choice of α in a small range around it. Note that since we have recovered the discriminative features, the simple classifier described above suffices for competitive classification performance. Such simple classifiers enables fast inference at test time. In addition, we employ Gaussian kernel SVM as classifiers and compare the results. We note that Gaussian kernel SVM provide better classification performance compared to linear SVM.

Next, we provide a method for estimating higher order score functions using auto-encoders. These higher-order score functions capture “local manifold structure” of the pdf. While the first-order score function is a vector (for a vector input), the higher-order functions are matrices and tensors, and thus capture richer information about the input distribution, which results in extracting better discriminative information.

3.1 Score function approximation through auto-encoders

In this section we provide a provable method for estimating higher order score functions by using DAE. Let $p(x)$ denote the joint probability density function of random vector $x \in \mathbb{R}^{d_x}$. We denote $\mathcal{S}_m(x)$ as the m-th order score function, which from (Janzamin et al., 2014) is given by Equation (1).

Lemma 1. *The m-th order score function can be estimated using DAE as follows*

$$\mathcal{S}_1(x) = \lim_{\sigma\to 0} \frac{1}{\sigma^2}\left(r^*_\sigma(x) - x\right), \tag{3a}$$

$$\mathcal{S}_m(x) = -\mathcal{S}_{m-1}(x) \otimes \nabla_x \log p(x) - \nabla_x \mathcal{S}_{m-1}(x). \tag{3b}$$

with $\mathcal{S}_0(x) = 1$.

Here $\otimes$ denotes the tensor product. Equation $(3a)$ is a direct result of (2) by (Alain and Bengio, 2012). Equation $(3b)$ is from (Janzamin et al., 2014).

Given $r^*(x)$, we can form $\mathcal{S}_1(x)$ from equation $3a$. For $m \geq 2$ we combine the above result with Equation $(3b)$. It is straight forward to see that

$$\mathcal{S}_2(x) = \mathcal{S}_1(x) \otimes \mathcal{S}_1(x) + \lim_{\sigma\to 0}\left[\frac{1}{\sigma^2}\left(\nabla_x r^*_\sigma(x) - I\right)\right], \tag{4a}$$

$$\mathcal{S}_3(x) = \mathcal{S}_1(x)^{\otimes 3} + \lim_{\sigma \to 0} \left[\frac{1}{\sigma^2} \operatorname{Sym}\left[(\nabla_x r^*_\sigma(x) - I) \otimes \mathcal{S}_1(x) \right] - \frac{1}{\sigma^2} \nabla_x^{(2)} r^*_\sigma(x) \right]. \tag{4b}$$

where the $\operatorname{Sym}(\cdot)$ subscript in the third term of $\mathcal{S}_3(x)$ refers to considering all transpositions of the corresponding tensor. Tensor transposition is defined in Appendix A. We need to mention that the approach proposed here can be extended to $m > 3$ as well.

3.2 Estimation of $\mathcal{S}_2(x)$ and $\mathcal{S}_3(x)$ for sigmoid activation function

Let $x \in \mathbb{R}^{d_x}$ and $\sigma(x)$ denote the sigmoid function applied to each entry of the vector x, i.e., entry i is computed as $[\sigma(x)]_i = \frac{1}{1+e^{-x_i}}$. We consider the auto-encoder wherein the reconstruction is defined as:

$$r(x) = \sigma\left(W\sigma\left(W^\top x + b_h \right) + b_v \right)$$

where $W \in \mathbb{R}^{d_x \times d_h}$ is the weight matrix and b_v and b_h are the biases for the visible and hidden units. Lemma 1, provides the means for obtaining the score functions of the auto-encoder model. We recall the first two derivatives of the sigmoid function applied to a scalar input z:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)), \quad \sigma''(z) = \sigma(z)(1 - \sigma(z))(1 - 2\sigma(z)).$$

Using the above, we may compute

$$\nabla_x r(x) = \operatorname{diag}_2[q]\left(I^\top, W \operatorname{diag}_2[p] W^\top \right), \tag{5}$$

$$\begin{aligned} \nabla_x^{(2)} r(x) = {} & \operatorname{diag}_3[\tilde{q}]\left(I^\top, W \operatorname{diag}_2[p] W^\top, W \operatorname{diag}_2[p] W^\top \right) \\ & + \operatorname{diag}_3[\tilde{p}]\left(\left[\operatorname{diag}_2[q] W\right]^\top, W^\top, W^\top \right), \end{aligned} \tag{6}$$

where for a given vector x, we define the diagonal matrix $\operatorname{diag}_2(x)_{ij} := x_i\ \delta(i = j)$, the diagonal tensor $\operatorname{diag}_3(x)_{ijk} := x_i\ \delta(i = j = k)$, vector $p := \sigma'\left(W^\top x + b_h\right)$, vector $q := \sigma'\left(W\sigma\left(W^\top x + b_h\right) + b_v\right)$, vector $\tilde{p} := \sigma''\left(W^\top x + b_h\right)$ and vector $\tilde{q} := \sigma''\left(W\sigma\left(W^\top x + b_h\right) +b_v\right)$.

Factor form for the cross-moment tensors: Note that computing and storing the cross-moment tensor $\mathbb{E}\left[y \otimes \mathcal{S}_3(x)\right]$ can be enormously expensive for high-dimensional problems. But, we do not run into this problem since we do not form the tensor. When score function is estimated via a DAE, the cross-moment tensor has an efficient factor form, and we can directly manipulate the factors to obtain the gradients through *multi-linear* operations, leading to efficient computational complexity. We establish the exact factor form of the empirical moment tensor $\hat{\mathbb{E}}\left[y \otimes \mathcal{S}_3(x)\right]$ using equations (4b),(6). Specifically, we establish that $\hat{T} := \hat{\mathbb{E}}[y \otimes \mathcal{S}_3(x)] = \sum_i y_i \otimes z_i \otimes z_i \otimes z_i, \quad z_i := g(x_i)$, where z_i is a closed-form function of x_i, and depends on the weights and biases of the auto-encoder.

Computational Complexity: If score function is estimated through DAE, the parallel computational complexity of FEAST framework is $\mathcal{O}(\log \min(d_h, d_x))$ per iteration with $\mathcal{O}(k d_x^2 d_h / \log(\min(d_h, d_x)))$ processors, where k, d_x and d_h denote rank of the cross-moment tensor (i.e. number of extracted features), input dimension, and the number of neurons in the DAE, respectively. Note that the dominant step in terms of computational complexity is the tensor decomposition step.

3.3 Score function properties

Theorem 2 (Higher order differential operators, *informal statement* (Janzamin et al., 2014))**.** *For random vector x, let $p(x)$ and $\mathcal{S}_m(x)$ respectively denote the joint density function and the corresponding m-th order score function in* (1)*. Then, under some mild regularity conditions, for all random variables y, we have*

$$\mathbb{E}\left[y \otimes \mathcal{S}_m(x)\right] = \mathbb{E}\left[\nabla_x^{(m)} G(x)\right],$$

where $\nabla_x^{(m)}$ denotes the m-th order derivative w.r.t. variable x and $G(x) := \mathbb{E}[y|x]$.

Thus, Janzamin et al. (2014) prove that the cross-moments between the label and the score function features are equal to the expected derivative of the label as a function of the input. But when are these derivatives informative? Indeed, in trivial cases, where the derivatives of the label function vanish over the support of the input distribution, these moments carry no information. Yet, such cases are pathological, as either there is no variation in the label function or the input distribution is nearly degenerate. Another possibility is that a certain derivative vanishes, when averaged over the input distribution, even though it is not zero everywhere. Nevertheless, the next derivative cannot be averaged out to zero, and will carry information about the variations of the label function. Hence, in practical scenarios, the cross-moments contain discriminative information.

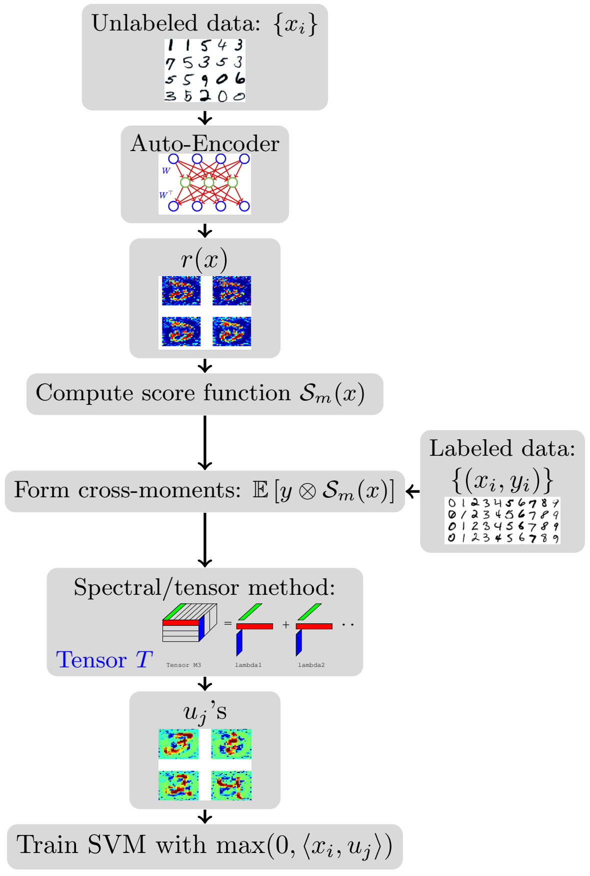
Unlabeled data: $\{x_i\}$
Auto-Encoder
$r(x)$
Compute score function $\mathcal{S}_m(x)$
Labeled data: $\{(x_i, y_i)\}$
Form cross-moments: $\mathbb{E}[y \otimes \mathcal{S}_m(x)]$
Spectral/tensor method:
Tensor T
u_j's
Train SVM with $\max(0, \langle x_i, u_j \rangle)$

Figure 3: Applying FEAST (Janzamin et al., 2014) to MNIST

4. Experiment Results on MNIST Data Set

We use MNIST dataset to empirically illustrate the practicality of the discriminative features obtained from FEAST in classification and provide more intuition. The overall pipeline can be summarized in five main steps. (1) Train a DAE model using unlabeled input samples. (2) Use the auto-encoder parameters to compute the score function and form the cross-moment between labels and the score function using a portion of labeled training samples. (3) Extract discriminative features via tensor decomposition. (4) Project rest of the training

Table 1: Accuracy for different estimated rank of the cross-moment $\hat{\mathbb{E}}[y \otimes \mathcal{S}_3(x)]$ (estimated rank is the same as # of discriminative features).

Rank	50	200	500	1000
Accuracy (Classifier: Linear SVM)	86.5%	90.65%	94.4%	95.6%
Accuracy (Classifier: Gaussian Kernel SVM)	95.88%	96.12%	96.86%	97.54%

samples into the new feature space and train a classifier in this space. (5) Report the classification accuracy on test set. The end-to-end framework and the input/output at each step are depicted in Figure 3. The details of the implementation steps are provided in Appendix B.2. Table 1 summarizes the classification accuracy on the test set.

Order of score function: Note that for MNIST the cross-moment $\mathbb{E}[y \otimes \mathcal{S}_1(x)]$ is a matrix. Thus, the rank is bounded by the number of labels, 10. This limits the number of discriminative features that could be obtained to 10. However, if we use higher-order score functions $\mathcal{S}_m$, $m \geq 2$, the cross-moment is a tensor. Hence, the rank can exceed the dimension and we obtain overcomplete representations. The extracted discriminative features when using the first-order score function and third-order score function for the cross-moment are shown in Figures 4, 5. It can readily be seen that although the former does not provide good discriminative features, the latter reveals features that distinguish digits.

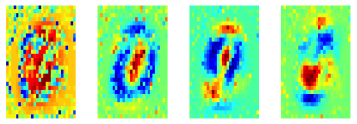
Figure 4: Leading four right singular vectors of $\hat{\mathbb{E}}[y \otimes \mathcal{S}_1(x)]$. The result is not satisfactory.

Some observations in MNIST: (a) Increasing the number of hidden neurons in DAE does not improve the results of the subsequent steps noticeably, while it makes the computation more expensive. We use 100 neurons. (b) The number of extracted discriminative features, size of the hidden layer of the DAE, batch size used to train the DAE and regularization parameter in SVM affect the classification performance. The most effective parameter is the number of extracted discriminative features. Note that the number of discriminative features is equal to the cross-moment tensor rank. Table 1 summarizes the obtained accuracy for different estimates of the rank. We obtain 95.6% accuracy with rank 1000 using a linear SVM and 97.54% using a non-linear SVM with Gaussian kernel SVM. Intuitively, by increasing the rank, we retrieve more discriminative directions and we train the classifier with a higher dimensional input. Hence the points are more separable by the classifier. (c) Using a specific nonlinear gating function such as $\max(0, U^\top x_i)$ affects the classification accuracy marginally compared to effect of other parameters mentioned above. We did not tune our steps extensively and used a simple classifier. Yet, we achieve promising results due to extracting good discriminative features.

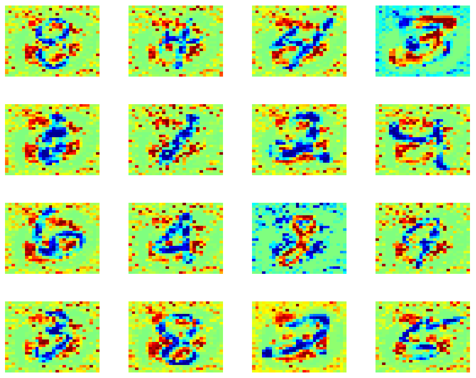
Figure 5: Sixteen discriminative features obtained by tensor decomposition on tensor $\hat{\mathbb{E}}[y \otimes \mathcal{S}_3(x)]$. Derived features can be used to distinguish digits/groups.

References

Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data generating distribution. *arXiv preprint arXiv:1211.4246*, 2012.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor Methods for Learning Latent Variable Models. *J. of Machine Learning Research*, 15:2773–2832, 2014a.

Anima Anandkumar, Rong Ge, and Majid Janzamin. Guaranteed Non-Orthogonal Tensor Decomposition via Alternating Rank-1 Updates. *arXiv preprint arXiv:1402.5180*, Feb. 2014b.

Anima Anandkumar, Rong Ge, and Majid Janzamin. Sample Complexity Analysis for Learning Overcomplete Latent Variable Models through Tensor Methods. *arXiv preprint arXiv:1408.0553*, Aug. 2014c.

Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015. URL `http://www.sandia.gov/~tgkolda/TensorToolbox/`.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1872–1886, 2013.

Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Fabio Gagliardi Cozman, Ira Cohen, and M Cirelo. Unlabeled data can degrade classification performance of generative classifiers. In *FLAIRS Conference*, pages 327–331, 2002.

Tommi Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Score Function Features for Discriminative Learning: Matrix and Tensor Frameworks. *arXiv preprint arXiv:1412.2863*, Dec. 2014.

Nikos Karampatziakis and Paul Mineiro. Discriminative features via generalized eigenvectors. In *Proceedings of The 31st International Conference on Machine Learning*, pages 494–502, 2014.

Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng. ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. In *NIPS*, pages 1017–1025, 2011.

Laurens Maaten. Learning discriminative fisher kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 217–224, 2011.

Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

Hua Wang, Feiping Nie, and Heng Huang. Robust and discriminative self-taught learning. In *Proceedings of The 30th International Conference on Machine Learning*, pages 298–306, 2013.

Appendix A. Notations and Tensor Preliminaries

Tensor as multilinear form: We view a tensor $T \in \mathbb{R}^{d\times d\times d}$ as a multilinear form. Consider matrices $M_l \in \mathbb{R}^{d\times d_l}, l \in \{1,2,3\}$. Then tensor $T(M_1, M_2, M_3) \in \mathbb{R}^{d_1} \otimes \mathbb{R}^{d_2} \otimes \mathbb{R}^{d_3}$ is defined as

$$T(M_1, M_2, M_3)_{i_1,i_2,i_3} := \sum_{j_1,j_2,j_3\in[d]} T_{j_1,j_2,j_3} \cdot M_1(j_1, i_1) \cdot M_2(j_2, i_2) \cdot M_3(j_3, i_3). \tag{7}$$

In particular, for vectors $u, v, w \in \mathbb{R}^d$, we have [2]

$$T(I, v, w) = \sum_{j,l\in[d]} v_j w_l T(:, j, l) \ \in \mathbb{R}^d, \tag{8}$$

which is a multilinear combination of the tensor mode-1 fibers. Similarly $T(u, v, w) \in \mathbb{R}$ is a multilinear combination of the tensor entries, and $T(I, I, w) \in \mathbb{R}^{d\times d}$ is a linear combination of the tensor slices.

CP decomposition and tensor rank: A 3rd order tensor $T \in \mathbb{R}^{d\times d\times d}$ is said to be rank-1 if it can be written in the form

$$T = w \cdot a \otimes b \otimes c \Leftrightarrow T(i, j, l) = w \cdot a(i) \cdot b(j) \cdot c(l), \tag{9}$$

where notation $\otimes$ represents the *tensor (outer) product*, and $a \in \mathbb{R}^d$, $b \in \mathbb{R}^d$, $c \in \mathbb{R}^d$ are unit vectors (without loss of generality). A tensor $T \in \mathbb{R}^{d\times d\times d}$ is said to have a CP rank $k \geq 1$ if it can be written as the sum of k rank-1 tensors

$$T = \sum_{i\in[k]} w_i a_i \otimes b_i \otimes c_i, \quad w_i \in \mathbb{R},\ a_i, b_i, c_i \in \mathbb{R}^d. \tag{10}$$

With the third order score function ($m = 3$), we carry out tensor decomposition of the form $\sum_i \lambda_i u_i \otimes u_i \otimes u_i$, as depicted in Figure 7.

Finally, the transposition of a tensor with respect to a permutation matrix is defined as follows.

2. Compare with the matrix case where for $M \in \mathbb{R}^{d\times d}$, we have $M(I, u) = Mu := \sum_{j\in[d]} u_j M(:, j) \in \mathbb{R}^d$.

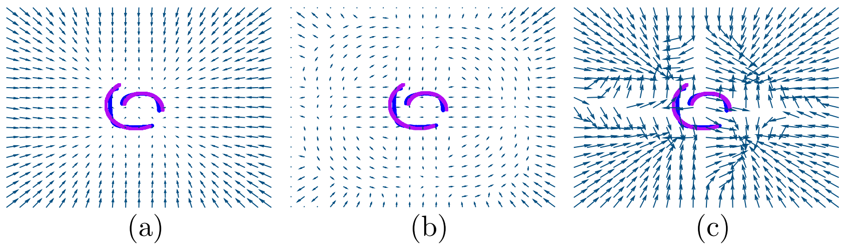

Figure 6: Zoomed out version: Reconstruction of Swiss roll using denoising auto-encoder (DAE) and extracting discriminative directions using proposed method FEAST. In (a), (b), (c), we plot score functions $\mathcal{S}_m(x)$ approximated using DAE. (a) Vector field $\mathcal{S}_1(x)$. (b) Top eigenvector of matrix field $\mathcal{S}_2(x)$. (c) Top eigenvector of tensor field $\mathcal{S}_3(x)$.

Definition 3 (Tensor transposition)**.** *Consider tensor $A \in \bigotimes^r \mathbb{R}^d$ and permutation vector $\pi = [\pi_1, \pi_2, \dots, \pi_r] \in \mathbb{R}^r$ as a permutation of index vector $1 : r$. Then, the π-transpose of A denoted by $A^{\langle \pi \rangle}$ is defined such that it satisfies*

$$A^{\langle \pi \rangle}(j_{\pi_1}, \dots, j_{\pi_r}) = A(j_1, \dots, j_r).$$

In other words, the i-th mode of tensor $A^{\langle \pi \rangle}$ corresponds to the π_i-th mode of tensor A.

Appendix B. Implementation Details

B.1 Visualizing higher order score function for Swiss roll data set

Following the existing convention, we generate the Swiss roll as follows. We generate 4000 uniformly-spaced points ζ such that $\zeta \in ([5, 8] \cup [9, 12])$ and half of the point are chosen from each segment. Then we transform each point ζ using the map $x = \phi(\zeta) = (\zeta \cos \zeta, \zeta \sin \zeta)$. We consider a 2-class classification where points from the first and second segment are assigned the label $+1, -1$ respectively. The results on estimating the score functions of order $1, 2$ are shown in Figure 1 in the main text. Figure 6 provides the zoomed out version for estimating the score functions of order $1, 2$ and 3 .

Next we provide detailed explanation on implementation steps for MNIST data set.

B.2 Implementation details for MNIST data set

Step 1: First, we train a denoising auto-encoder[3] with input having 20% random corruption drawn from the original MNIST database to obtain the weight matrix and bias vector for the input layer of the DAE. i.e., the goal is to find W and b for the input layer where the transfer function is $f(x) = \sigma(Wx + b)$. Note that we vectorize each image sample (which is 28×28) , so $d_x = 784$ and we set $d_h = 50$ for the hidden layer. We train the DAE using stochastic gradient descent for 100 epochs and using minibatches of size 15, with a learning rate of 0.1.

Step 2: We use a subset [4] of the original MNIST dataset for the rest of our pipeline. For each digit 1000 samples are available. We select 300 samples of each digit to perform the

3. We use the code available at `http://deeplearning.net/tutorial/dA.html`
4. The dataset is available at `http://cis.jhu.edu/~sachin/digit/digit.html`

feature extraction using the score function method. We then compute the third order score function using each sample. To form the cross-moments with the labels, we use the one-hot encoding method for the labels. Note that the result is a fourth-order tensor. The first mode bears no discriminative information as the formed tensor is of the form $\sum_i e_i \otimes u_i^{\otimes 3}$ where e_i is the basis vector. Therefore, we discard the first mode after tensor decomposition.

Step 3: For MNIST dataset we perform stochastic ALS updates. We perform the tensor decomposition in a 'stochastic manner' with 'deflation', described as follows. We form $y_i \otimes S_3(x_i)$ for each labeled sample (x_i, y_i) and perform four iterations of the ALS algorithm for a rank-1 CP decomposition[5] described in Appendix A. This tensor decomposition method is explained in more details in Appendix B.3. We use random initialization for the first iteration, i.e., when $i = 1$. Afterwards, for sample i, we initialize the ALS algorithm with the factor matrices obtained from the ALS iterations executed on sample $i-1$. We repeat this 20 times to get a new discriminative feature. Let the output tensor be T. Then, we continue this process on $[y_i \otimes S_3(x_i) - T]$. As mentioned earlier, this process is called 'deflation'. In order to find 1000 discriminative features, we perform deflation 1000 times and collect all the rank-1 components. Hence, we obtain 1000 discriminative features. The key point to note is that we never form the $10 \times 784 \times 784 \times 784$ tensor. This is because we perform tensor decomposition in a stochastic manner with deflation. Also, we note that this is much faster and efficient in terms of memory requirements than exactly forming the tensor and doing the decomposition.

Step 4: For projection into the new feature space, we note that the tensor obtained from Step 3 is symmetric in the last three modes and the first mode is ignored since it bears no information as explained above. Let U be the 784×1000 feature matrix. We compute $U^\top x_i$ for each input sample from the the training set marked for the SVM and perform $\max(0, U^\top x_i)$ where the max operator is applied elementwise. The justification for this non-linear gating is its robustness property (Karampatziakis and Mineiro, 2014). Then, we use 500 samples to train a multiclass SVM[6]. The best performance is obtained for the regularization parameter $C = 1$. We have also illustrated the extracted discriminative features when using the first-order score function for the cross-moment, see Figure 4. Note that the rank of $\mathbb{E}[y \otimes \mathcal{S}_1(x)]$ can atmost be 10, hence this limits the number of disctiminative features that could be obtained. However, by going to higher-order score functions $\mathcal{S}_m$, $m \geq 2$, we may obtain overcomplete representations. In this work we have considered $\mathcal{S}_3$ extensively.

Step 5: We evaluate the performance by computing the classification accuracy on the remaining 100 samples. Table 1 summarizes the obtained accuracy for different ranks.

B.3 Stochastic tensor decomposition: stochastic rank-1 ALS updates

When using stochastic rank-1 ALS updates, we obtain the components of the empirical cross-moment tensor $\hat{T}$ using a stochastic version of the popular rank-1 alternating minimization. For simplicity, we assume that the label is a scalar, so that $\hat{T}$ is a third order tensor. The method can be directly extended to more complex outputs. The rank-1 ALS

5. We use the MATLAB tensor toolbox (Bader et al., 2015)
6. We use Liblinear library for MATLAB available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

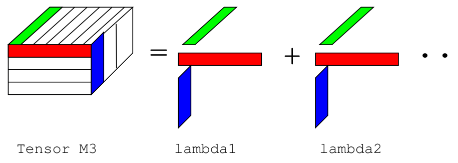

Figure 7: Tensor decomposition into rank-1 elements

method attempts to minimize the objective

$$\min_{a,b,c} \left\| \hat{T} - a \otimes b \otimes c \right\|_{\mathbb{F}},$$

where $\| \cdot \|_{\mathbb{F}}$ denotes the Frobenious norm. The ALS method updates each vector by fixing the other two, e.g., a is updated by fixing b and c. The update has a closed form as

$$a \leftarrow \frac{\hat{T}(I,b,c)}{\|\hat{T}(I,b,c)\|_2},$$

where $\hat{T}(I,b,c) = \sum_{j,l} \hat{T}(:,j,l) b_j c_l$.

As explained in Section 3, when score function is estimated via a DAE, we form an efficient factor form of the cross-moment tensor using samples, and we can manipulate the samples directly to perform the tensor decomposition steps as *multi-linear* operations, leading to efficient computational complexity. We establish the exact factor form of the empirical tensor $\hat{\mathbb{E}}\left[y \otimes \mathcal{S}_3(x)\right]$ in Equations ($4b$) and (6). Specifically, we establish that

$$\hat{T} := \hat{\mathbb{E}}[y \otimes \mathcal{S}_3(x)] = \sum_i y_i \otimes z_i \otimes z_i \otimes z_i, \quad z_i := g(x_i), \tag{11}$$

where z_i is a function of x_i (which we obtain in closed form), and depends on the weights and biases of the auto-encoder. Note that when $\hat{T}$ is the empirical cross-moment tensor in (11), $\hat{T}(I,b,c) = \sum_i y_i \cdot \langle z_i, b\rangle\langle z_i, c\rangle z_i$ can be computed efficiently, as it only consists of multi-linear operations. The stochastic version of this method only uses samples in the current mini-batch to compute the update. In our experiments, we used a single sample as the mini-batch, and found that it gave good performance, in addition to being fast to implement. We run this update to convergence. Once we learn a component, we subtract it from the tensor and run tensor decomposition on the result assuming now the rank is less by 1. This procedure is called deflation. Then we repeat the rank-1 ALS. In the end, we can appropriately symmetrize the estimated components to obtain the set of vectors $\{u_i\}$.