# Generalization in Unsupervised Learning

Karim T. Abou-Moustafa and Dale Schuurmans

Department of Computing Science, University of Alberta
Edmonton, Alberta T6G 2E8, CANADA
{aboumous,daes}@ualberta.ca

**Abstract.** We are interested in the following questions. Given a finite data set $\mathcal{S}$, with neither labels nor side information, and an unsupervised learning algorithm $\mathsf{A}$, can the generalization of $\mathsf{A}$ be assessed on $\mathcal{S}$? Similarly, given two unsupervised learning algorithms, $\mathsf{A}_1$ and $\mathsf{A}_2$, for the same learning task, can one assess whether one will generalize "better" on future data drawn from the same source as $\mathcal{S}$? In this paper, we develop a general approach to answering these questions in a reliable and efficient manner using mild assumptions on $\mathsf{A}$. We first propose a concrete generalization criterion for unsupervised learning that is analogous to prediction error in supervised learning. Then, we develop a computationally efficient procedure that realizes the generalization criterion on finite data sets, and propose and extension for comparing the generalization of two algorithms on the same data set. We validate the overall framework on algorithms for clustering and dimensionality reduction (linear and nonlinear).

## 1 Introduction

The goal of unsupervised learning is to autonomously capture and model latent relations among the variables of a data set. Such latent relations are usually in the form of regularities and statistical dependencies known as *the underlying structure of the data distribution*. Unlike supervised learning, there are no desired target answers to guide and correct the learning process. However, similar to supervised learning, unsupervised learning algorithms generate estimates that are functions of sample data drawn from an unknown distribution $\mathscr{P}$. As such, it is natural to ask questions related to the generalization capability of these estimates, as well as questions on the choice of these estimates (model selection) [11].

In supervised learning, questions of generalization have been scrutinized, equally, in theory and in practice; see for instance [8, 15, 22, 9, 14, 5, 6, 17, 20, 24]. In unsupervised learning, however, few efforts have acknowledged and addressed the problem in general. For instance, [11] approximates the expected loss of finite parametric models such as principle component analysis (PCA) and $k$-Means clustering based on asymptotic analysis and central limit results.

One possible reason for the scarcity of such efforts is the subjective nature of unsupervised learning, the diversity of tasks covered (such as clustering, density estimation, dimensionality reduction, feature learning, etc.), and the lack of a unified framework that incorporates a significant subset of these tasks. Another reason is that the principles underlying supervised learning are often distinct from those underlying unsupervised learning. In supervised learning, the final result of a learning algorithm is a function $f^*$ that minimizes the expected loss (possibly plus a regularizer) under the unknown true distribution $\mathscr{P}$, which can be applied to new points not included during training. Since $\mathscr{P}$ is unknown, the learning algorithm selects $f^*$ that minimizes an empirical average of the loss as a surrogate for the expected loss. Therefore, since the loss measures the difference between the estimated and expected outputs, its average provides an indicator of generalization error. The validity of this mechanism, however, rests on (*i*) the existence of target outputs, and (*ii*) consistency of the empirical average of the loss [22].

In the unsupervised learning, the characterization is different. First, the target output is not available. Second, an unsupervised learning algorithm $\mathsf{A}$ produces an output that is a re-representation of the input; hence loss functions in this setting usually assess a reconstruction error between the output and input [25]. Third, there are various unsupervised learning algorithms that do not minimize a reconstruction error yet still produce an output that is a re-representation of the input: see for example the recent literature on moments-based methods for latent variable models and finite automata [12, 21, 2, 1].

These observations motivate us to deal with unsupervised learning algorithms in an abstract form. In particular, we consider an unsupervised learning algorithm $\mathsf{A}$ as an abstract function – a black box – that maps an input $x$ to an output $y$. The advantage of this view is that (*i*) it is independent of the learning task, and (*ii*) it provides a simple unified view for these algorithms without being overly dependent on internal details.

Based on this perspective, we propose a general definition for generalization of an unsupervised learning algorithm on a data set $\mathcal{S}$. The framework is based on a general loss function $\ell$ that measures the reconstruction error between the input and output of $\mathsf{A}$, which is not necessarily the loss minimized by $\mathsf{A}$ (if any). To study the generalization of $\mathsf{A}$ under the black box assumption and an external loss $\ell$, we will assume that $\mathsf{A}$ satisfies a certain notion of algorithmic stability under some mild assumptions on $\ell$. Given this notion of stability, we derive a finite useful upper bound on $\mathsf{A}$'s expected loss, which naturally lends itself to a generalization criterion for unsupervised learning. As a second contribution, we develop an efficient procedure to realize this generalization criterion on finite data sets, which can be extended to comparing the generalization of two different unsupervised learning algorithms on a common data source. Finally, we apply this generalization analysis framework and evaluation procedure to two unsupervised learning problems; clustering and dimensionality reduction.

### 1.1 Preliminaries and Setup

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^k$ be the input and output spaces, respectively. [1] Let $\mathcal{S} \in \mathcal{X}^n$ be a training set of size $n$ drawn IID from an unknown distribution $\mathscr{P}_{\mathbf{x}}$ defined on a measurable space $(\mathcal{X}, \Sigma)$ with domain $\mathcal{X}$ and $\sigma$-algebra $\Sigma$. We denote this as $\mathcal{S} \sim \mathscr{P}_{\mathbf{x}}$ where $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n$. For each $\mathbf{x}_i \in \mathcal{S}$ there is a corresponding output $\mathbf{y}_i$, $1 \leq i \leq n$, with appropriate dimension $k$. For convenience, $\mathcal{S}$ can be represented as a matrix $\mathbf{X}_{n\times d}$, while the output can also be represented as a matrix $\mathbf{Y}_{n\times k}$.

[1] Notation: Lower case letters $x, m, i$ denote scalars and indices. Upper case letters $X, Y$ denote random variables. Bold lower case letters $\mathbf{x}, \mathbf{y}$ denote vectors. Bold upper case letters $\mathbf{A}, \mathbf{B}$ are matrices. Distributions $\mathscr{P}, \mathscr{G}$ will be written in script. Calligraphic letters $\mathcal{X}, \mathcal{Y}$ denote sets.

An unsupervised learning algorithm $\mathsf{A}$ is a mapping from $\mathcal{X}^n$ to the class of functions $\mathcal{F}$ s.t. for $f \in \mathcal{F}$, $f : \mathcal{X} \to \mathcal{Y}$. Thus, $\mathsf{A}$ takes as input $\mathcal{S}$, selects a particular $f^*$ from $\mathcal{F}$, and estimates an $n \times k$ output matrix $\widehat{\mathbf{Y}} \equiv \mathsf{A}_{\mathcal{S}}(\mathbf{X})$, or $\widehat{\mathbf{y}} \equiv \mathsf{A}_{\mathcal{S}}(\mathbf{x})$, [2] where $\mathsf{A}_{\mathcal{S}}$ denotes the output of $\mathsf{A}$ (i.e. $f^* \in \mathcal{F}$) after training on $\mathcal{S}$. The algorithm $\mathsf{A}$ could also have certain parameters, denoted $\boldsymbol{\theta}_{\mathsf{A}}$, that the user can tune to optimize its performance. We assume that $\mathsf{A}$ and its output functions in $\mathcal{F}$ are all measurable maps.

## 2 A General Learning Framework

The problem of unsupervised learning is that of selecting a function $f^* \in \mathcal{F}$ that transforms input $\mathbf{x}$ into an output $\widehat{\mathbf{y}} \equiv \mathsf{A}_{\mathcal{S}}(\mathbf{x})$ in some desired way. Here we assume that $\mathsf{A}$ is a black box that takes $\mathcal{S}$ and produces a map $f^*$ from $\mathbf{x}$ to $\widehat{\mathbf{y}}$. Since we are ignoring $\mathsf{A}$'s internal details, assessing its generalization requires us to consider an additive *external loss* function $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ that measures the reconstruction error between $\mathbf{x}$ and $\widehat{\mathbf{y}}$. Thus, the expected loss for $\mathsf{A}_{\mathcal{S}}$ with respect to $\ell$ is defined as:

$$R(\mathsf{A}_{\mathcal{S}}) \equiv \mathbb{E}\left[\ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x})\right] = \int \ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x})) d\mathscr{P}_{\mathbf{x}}. \tag{1}$$

Unfortunately $R(\mathsf{A}_{\mathcal{S}})$ cannot be computed since $\mathscr{P}_{\mathbf{x}}$ is unknown, and thus it has to be estimated from $\mathcal{S} \in \mathcal{X}^n$. A simple estimator for $R(\mathsf{A}_{\mathcal{S}})$ is the empirical estimate:

$$\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{x}_i, \mathsf{A}_{\mathcal{S}}(\mathbf{x}_i)). \tag{2}$$

To obtain a practical assessment of the generalization of $\mathsf{A}$, we need to derive an upper bound for the quantity $\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) - R(\mathsf{A}_{\mathcal{S}})$. Given the generality of this setting, one needs to resort to worst case bounds. However, this cannot be done without introducing additional assumptions about the behaviour of $\mathsf{A}$. For example, if one assumes that $\mathsf{A}$ chooses its output from a class of functions $\mathcal{F}$ such that the class of loss random variables $\varLambda : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}+$ induced by $\mathcal{F}$, i.e. $\varLambda = \ell \circ \mathcal{F}$, is uniformly upper bounded by $c < \infty$ and VCdim$(\varLambda) = h < \infty$, then with probability at least $1 - \eta$ there is a uniform concentration of $\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}})$ around $R(\mathsf{A}_{\mathcal{S}})$:

$$R(\mathsf{A}_{\mathcal{S}}) \leq \widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) + \frac{\tau c}{2}\left(1 + \sqrt{1 + \frac{4\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}})}{\tau c}}\right), \tag{3}$$

where $\tau = 4n^{-1}\left[h(\ln 2n/h + 1) - \ln \eta\right]$ [22, 23]. Rademacher or Gaussian complexities can also be used to obtain similar concentration inequalities [3]. The caveat is that such an analysis is worst case and the resulting bounds, such as (3), are too loose to be useful in practice. This suggests that we need to make stronger assumptions on $\mathsf{A}$ to achieve more useful bounds on the quantity $\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) - R(\mathsf{A}_{\mathcal{S}})$.

[2] For example, in $k$-Means clustering, the elements of $\widehat{\mathbf{Y}}$ could be the corresponding cluster centers assigned to each $\mathbf{x}_i$ from a set of $k$ such centers. In nonlinear dimensionality reduction, the output could be the $n \times n$ low rank matrix $\widehat{\mathbf{Y}}$. In density estimation using a mixture model, $\mathsf{A}$ could output the $n \times 1$ matrix $\mathbf{Y}$ with the density value of each $\mathbf{x}_i$.

## 2.1 Generalization and Stability

To achieve a more practical criterion and assessment procedure, we need to introduce some form of additional assumptions on $\mathsf{A}$ without sacrificing too much generality. To this end, we investigate an assumption that $\mathsf{A}$ satisfies a particular notion of algorithmic stability that allows us to derive a more useful and a tighter upper bound on $\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) - R(\mathsf{A}_{\mathcal{S}})$. Algorithmic stability has been successfully applied in learning theory to derive generalization bounds for supervised learning algorithms, but has yet to be formally applied to unsupervised learning. Among the different notions of stability, the uniform stability of [5] is considered to be the strongest since it implies other notions of stability such as: hypothesis stability, error stability, point–wise hypothesis stability, everywhere stability, CVLOO stability, etc. [8, 14, 16, 17, 20].

To define uniform stability for $\mathsf{A}$ in the unsupervised learning context, we require the following definitions. For any $\mathcal{S} \in \mathcal{X}^n$, we define $\forall i$, $1 \le i \le n$, the modified training set $\mathcal{S}^{\setminus i}$ by removing from $\mathcal{S}$ the $i$-th element: $\mathcal{S}^{\setminus i} = \{\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_n\}$. We assume that $\mathsf{A}$ is symmetric with respect to $\mathcal{S}$; i.e. it does not depend on the elements' order in $\mathcal{S}$. Further, we require that the external loss $\ell$ be "*well behaved*" with respect to slight changes in $\mathcal{S}$; i.e. if $\varepsilon = \ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x}))$, $\varepsilon' = \ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}'}(\mathbf{x}))$, and $\mathcal{S}'$ is slightly different from $\mathcal{S}$ such that $\mathsf{A}_{\mathcal{S}}(\mathbf{x}) \approx \mathsf{A}_{\mathcal{S}'}(\mathbf{x})$, then the difference between $\varepsilon$ and $\varepsilon'$ should be small. The notion of "*well behaved*" is formally imposed by requiring that $\ell$ is Lipschitz continuous, and that $\mathsf{A}$ is uniformly $\beta$–stable with respect to $\ell$. This uniform $\beta$–stability is defined as follows:

**Definition 1 (Uniform $\beta$–Stability).** *An algorithm $\mathsf{A}$ is uniformly $\beta$–stable with respect to the loss function $\ell$ if for any $\mathbf{x} \in \mathcal{X}$, the following holds:*

$$\forall\, \mathcal{S} \in \mathcal{X}^n, \max_{i=1,\dots,n} |\ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x})) - \ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{x}))| \le \beta.$$

Note that $\beta$ is a function of $n$ and we assume that stability is non-increasing as a function of $n$. Hence, in the following, $\beta$ can be denoted by $\beta_n$.

**Definition 2 (Stable Algorithm).** *Algorithm $\mathsf{A}$ is stable if $\beta_n \propto \frac{1}{n}$.*[3]

The analogy between our definition of uniform $\beta$–stability and the uniform $\beta$–stability in supervised learning can be explained as follows. The uniform $\beta$–stability in [5] is in terms of $\ell(\mathsf{A}_{\mathcal{S}}, z)$ and $\ell(\mathsf{A}_{\mathcal{S}^{\setminus i}}, z)$, where $z = (\mathbf{x}, y)$, $\mathbf{x}$ is the input vector, and $y$ is its expected output (or true label). Note that $\ell(\mathsf{A}_{\mathcal{S}}, z)$ can be written as $\ell(f_{\mathcal{S}}(\mathbf{x}), y)$, where $f_{\mathcal{S}}$ is the hypothesis learned by $\mathsf{A}$ using the training set $\mathcal{S}$. Similarly, $\ell(\mathsf{A}_{\mathcal{S}^{\setminus i}}, z)$ can be written as $\ell(f_{\mathcal{S}^{\setminus i}}(\mathbf{x}), y)$. Observe that the difference between $\ell(f_{\mathcal{S}}(\mathbf{x}), y)$ and $\ell(f_{\mathcal{S}^{\setminus i}}(\mathbf{x}), y)$ is in the hypotheses $f_{\mathcal{S}}$ and $f_{\mathcal{S}^{\setminus i}}$. Note also that in supervised learning, the loss $\ell$ measures the discrepancy between the expected output $y$ and the estimated output $\widehat{y} = f_{\mathcal{S}}(\mathbf{x})$. In our unsupervised learning setting, the expected output is not available, and the loss $\ell$ measures the reconstruction error between $\mathbf{x}$ and $\widehat{\mathbf{y}} \equiv \mathsf{A}_{\mathcal{S}}(\mathbf{x})$. Hence, we replace $\ell(\mathsf{A}_{\mathcal{S}}, z)$ by $\ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x}))$, and $\ell(\mathsf{A}_{\mathcal{S}^{\setminus i}}, z)$ by $\ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{x}))$ to finally obtain Definition 1.

[3] $\beta_n \propto \frac{1}{n} \Longrightarrow \beta_n = \frac{\kappa}{n}$, for some constant $\kappa > 0$.

Note that the uniform $\beta$–stability of $\mathsf{A}$ with respect to $\ell$ is complimentary to the continuous Lipschitz condition on $\ell$. If $\mathsf{A}$ is uniformly $\beta$–stable, then a slight change in the input will result in a slight change in the output, resulting in a change in the loss bounded by $\beta$. The following corollary upper bounds the quantity $\widehat{R}_{\text{EMP}}(\mathsf{A}_\mathcal{S}) - R(\mathsf{A}_\mathcal{S})$ using the uniform $\beta$–stability of $\mathsf{A}$.

**Corollary 1.** *Let* $\mathsf{A}$ *be a uniformly $\beta$–stable algorithm with respect to $\ell$, $\forall\ \mathbf{x} \in \mathcal{X}$, and $\forall\ \mathcal{S} \in \mathcal{X}^n$. Then, for any $n \geq 1$, and any $\delta \in (0,1)$, the following bounds hold (separately) with probability at least $1-\delta$ over any $\mathcal{S} \sim \mathscr{P}_\mathbf{x}$:*

$$(i) \quad R(\mathsf{A}_\mathcal{S}) \leq \widehat{R}_{EMP}(\mathsf{A}_\mathcal{S}) + 2\beta + (4n\beta + c)\sqrt{\frac{\log(1/\delta)}{2n}}, \tag{4}$$

$$(ii) \quad R(\mathsf{A}_\mathcal{S}) \leq \widehat{R}_{LOO}(\mathsf{A}_\mathcal{S}) + \beta + (4n\beta + c)\sqrt{\frac{\log(1/\delta)}{2n}}, \quad where \tag{5}$$

$\widehat{R}_{LOO}(\mathsf{A}_\mathcal{S}) = \frac{1}{n}\sum_{i=1}^n \ell(\mathbf{x}_i, \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{x}_i))$*, is the leave-one-out (LOO) error estimate.*

*Discussion.* The generalization bounds in (4) and (5) directly follow from Theorem 12 in [5] for the regression case. The reason we consider $\mathsf{A}$ under the regression framework is due to our characterization of unsupervised learning algorithms in which we consider the output $\widehat{y} \in \mathbb{R}^k$ is a re-representation of the input $\mathbf{x} \in \mathbb{R}^d$. This, in turn, defined the form of the external loss $\ell$ as $\ell : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$. This characterization is very similar to the multivariate regression setting, and hence our reliance on Theorem 12 in [5]. Note that if $\beta \propto \frac{1}{n}$, then the bounds in (4) and (5) will be tight.

Corollary 1 is interesting in our context for a few reasons. First, it defines a generalization criterion for unsupervised learning algorithms in general: if $\mathsf{A}$ is uniformly $\beta$–stable with respect to $\ell$ on $\mathcal{S}$, then the bounds in (4) and (5) hold with high probability. Note that the bound in (4) is tighter than the one in (3). Second, the bounds for $\widehat{R}_{\text{EMP}}$ and $\widehat{R}_{\text{LOO}}$ are very similar. Various works have reported that $\widehat{R}_{\text{EMP}}$ is an optimistically biased estimate for $R$, while $\widehat{R}_{\text{LOO}}$ is almost an unbiased estimate [8, 14, 5].[4] Therefore, an advantage of uniform $\beta$–stability is that this discrepancy is mitigated. This also shows that stability based bounds are more suitable for studying algorithms whose empirical error remains close to the LOO error.

Second, this result also shows that to be uniformly stable, a learning algorithm needs to significantly depart from the empirical risk minimization principle that emphasizes the minimization of $\widehat{R}_{\text{EMP}}$. That is, a stable algorithm $\mathsf{A}$ might exhibit a larger error during training but this would be compensated by a decrease in complexity of the learned function. This characteristic is exactly what defines the effects of regularization. Therefore, the choice for uniform stability allows one to consider a large class of unsupervised learning algorithms, including those formulated as regularized minimization of an internal loss.

[4] The LOO error estimate over $n$ samples, $\widehat{R}_{\text{LOO}_n}$, is an unbiased estimate for $\widehat{R}_{\text{LOO}_{n-1}}$. Since in most interesting cases $\widehat{R}_{\text{LOO}_n}$ converges with probability one, the difference between $\widehat{R}_{\text{LOO}_n}$ and $\widehat{R}_{\text{LOO}_{n-1}}$ becomes negligible for large $n$ [7, Ch. 24].

## 3 Empirical Generalization Analysis

Although the previous section defines a general criterion for generalization in unsupervised learning, in practice this criterion requires assessing the uniform stability of $\mathsf{A}$ on a finite data set $\mathcal{S}$. The quantity of interest in the uniform stability criterion is $|\ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}}(\mathbf{x})) - \ell(\mathbf{x}, \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{x}))|$, which is the amount of change in the loss with respect to the exclusion of one data point $\mathbf{x}_i$ from $\mathcal{S}$. Taking expectations with respect to $\mathscr{P}_{\mathbf{x}}$ and replacing the expected loss with the empirical estimator, we have that:

$$\forall\, \mathcal{S} \in \mathcal{X}^n \qquad \max_{i=1,\dots,n} |\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) - \widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}^{\setminus i}})| \leq \beta_n. \tag{6}$$

This states that for a uniformly $\beta_n$–stable algorithm with respect to $\ell$ on $\mathcal{S}$, the change in the empirical loss due to the exclusion of one sample from $\mathcal{S}$ is at most $\beta_n$. In the finite sample setting, this will be:

$$\max_{i=1,\dots,n} \left| \frac{1}{n} \sum_{j=1}^{n} \ell(\mathbf{x}_j, \mathsf{A}_{\mathcal{S}}(\mathbf{x}_j)) - \frac{1}{n-1} \sum_{\substack{j=1 \\ j\neq i}}^{n} \ell(\mathbf{x}_j, \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{x}_j)) \right| \leq \beta_n. \tag{7}$$

Inequality (7) contains an unknown parameter $\beta_n$ which cannot be upper bounded without any further knowledge on $\mathsf{A}$. In fact, given the black box assumption on $\mathsf{A}$ and the absence of information on $\mathscr{P}_{\mathbf{x}}$, we cannot obtain a uniform upper bound on $\beta_n$. This suggests that $\beta_n$ needs to be estimated from the data set $\mathcal{S}$. Also, recall from Definitions 1 and 2 that if $\beta_n \propto 1/n$, then the generalization bounds in (4) and (5) will hold with high probability. These two requirements raise the need for two procedures; one to estimate $\beta_n$ at increasing values of $n$, and another one to model the relation between the estimated $\beta_n$'s and the values of $n$. However, to consider these two procedures for assessing $\mathsf{A}$'s generalization, we need to introduce a further mild assumption on $\mathsf{A}$. In particular, we need to assume that $\mathsf{A}$ does not change its learning mechanism as the sample size is increasing from $n$ to $n+1$ for any $n \geq 1$. Note that if $\mathsf{A}$ changes its learning mechanism based on the sample size, then $\mathsf{A}$ can have inconsistent trends of $\beta_n$ with respect to $n$ which makes it unfeasible to obtain consistent confidence bounds for $\widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) - R(\mathsf{A}_{\mathcal{S}})$. Therefore, we believe that our assumption is an intuitive one, and is naturally satisfied by most learning algorithms.

### 3.1 Estimating $\beta_n$ From a Finite Data Set

Inequality (7) might suggest a simple procedure for estimating $\beta_n$: (*i*) Compute $\widehat{\mathbf{Y}} = \mathsf{A}_{\mathcal{S}}(\mathbf{X})$. (*ii*) Set $\mathbf{X}' = \mathbf{X}$, hold out sample $\mathbf{x}_i$ from $\mathbf{X}'$, and compute $\widehat{\mathbf{Y}}' = \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{X}')$, and set $B_i = |n^{-1}\ell(\mathbf{X}, \widehat{\mathbf{Y}}) - (n-1)^{-1}\ell(\mathbf{X}', \widehat{\mathbf{Y}}')|$. (*iii*) Repeat step (*ii*) $n$ times to obtain $\{B_1 \dots, B_n\}$, and then set set $\widehat{\beta}_n = \max\{B_1 \dots, B_n\}$. The problem with this procedure is three–fold. First, note that in the finite sample setting, Inequality (7) cannot be evaluated for $\forall \mathcal{S} \in \mathcal{X}^n$ as required in Inequality (6). Note also that the sample maximum is a noisy estimate, and hence is not reliable. Second, the LOO estimate suggested above is computationally expensive since it requires invoking $\mathsf{A}$ for $n$ times. Third, using all $\mathbf{X}$ to learn $\widehat{\mathbf{Y}}$ will not reflect $\mathsf{A}$'s sensitivity to the randomness in the data. If $\mathsf{A}$

**Algorithm 1** Generalization Analysis for Algorithm $\mathsf{A}$.

```
1: Require: Algorithm $\mathsf{A}$ and its input parameters $\boldsymbol{\theta}_{\mathsf{A}}$, data set $\mathcal{S}$, loss function $\ell$, number of
   subsamples $m$, and the sizes of subsamples, $n_t$ s.t. $n_1 < n_2 < n_3 < \cdots < n_\tau$.
2: for $t = 1$ to $\tau$ do
3:     for $j = 1$ to $m$ do
4:         $\mathbf{X}_j \leftarrow$ draw $n_t$ samples uniformly from $\mathcal{S}$
5:         $\widehat{\mathbf{Y}}_j \leftarrow \mathsf{A}_{\mathcal{S}}(\mathbf{X}_j; \boldsymbol{\theta}_{\mathsf{A}})$
6:         $\boldsymbol{\Phi} \leftarrow$ hold out one random samples from $\mathbf{X}_j$
7:         $\mathbf{X}'_j \leftarrow \mathbf{X}_j \setminus \boldsymbol{\Phi}$
8:         $\widehat{\mathbf{Y}}'_j \leftarrow \mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{X}'_j; \boldsymbol{\theta}_{\mathsf{A}})$
9:         $R_j \leftarrow \frac{1}{n_1}\ell(\mathbf{X}_j, \widehat{\mathbf{Y}}_j)$
10:        $R'_j \leftarrow \frac{1}{n_1 - 1}\ell(\mathbf{X}'_j, \widehat{\mathbf{Y}}'_j)$
11:        $B_j = |R_j - R'_j|$
12:    end for
13:    $\widehat{\beta}_{n_t} = \text{median}\{B_1, \ldots, B_j, \ldots, B_m\}$
14: end for
15: Return: $\mathcal{B} = \{\widehat{\beta}_{n_1}, \ldots, \widehat{\beta}_{n_t}, \ldots, \widehat{\beta}_{n_\tau}\}$
```

easily gets stuck in local minima, or $\mathsf{A}$ has tendency to overfit the data, learning using all $\mathbf{X}$ will obscure such traits.

Our proposed procedure for estimating $\beta_n$, depicted in Algorithm 1, addresses the above issues in the following ways. First it is based on repeated random subsampling (with replacement) from the original data set $\mathcal{S}$, similar in spirit to bootstrapping [10]. Second, for each subsample of size $n_t$, the procedure obtains an estimate for the empirical loss before and after holding out one random sample. After repeating this subsampling process $m$ times, $m \ll n$, the procedure obtains one estimate for $\beta_n$, denoted by $\widehat{\beta}_{n_t}$, for sample size $n_t$. Note that $\widehat{\beta}_{n_t}$ is the median of $B_j$'s to increase the robustness of the estimate. This process is repeated $\tau$ times and the final output of Algorithm 1 is the set of $\widehat{\beta}_{n_t}$'s for the increasing values of $n_t$.

The proposed procedure is computationally intensive, yet it is efficient, scalable, and provides control over the accuracy of the estimates. First, note that the proposed procedure is not affected by the fact that $\mathsf{A}$ is an unsupervised learning algorithm. If $\mathsf{A}$ is a supervised learning algorithm, then assessing its generalization through uniform $\beta$–stability results will still require $2\tau m$ calls for $\mathsf{A}$, as it is the case for the unsupervised setting discussed here. Thus, the procedure does not impose a computational overhead given the absence of the expected output, and the black box assumption on $\mathsf{A}$. Second, considering scalability for large data sets, the procedure can be fully parallelized on multiple core architectures and computing clusters [13]. Note that in each iteration $j$ the processing steps for each subsample are independent from all other iterations, and hence all $m$ subsamples can be processed in an embarrassingly parallel manner. Note also that in each iteration, $\mathsf{A}_{\mathcal{S}}(\mathbf{X}_j)$ and $\mathsf{A}_{\mathcal{S}^{\setminus i}}(\mathbf{X}'_j)$ can also be executed in parallel.

Parameters $m$ and size of the subsamples, $n_1, n_2, \ldots, n_\tau$, control the tradeoff between computational efficiency and estimation accuracy. These parameters are user–specified and they depend on the data and problem in hand, its size $n$, $\mathsf{A}$'s complexity, and the available computational resources. Parameter $m$ needs to be sufficient to reduce

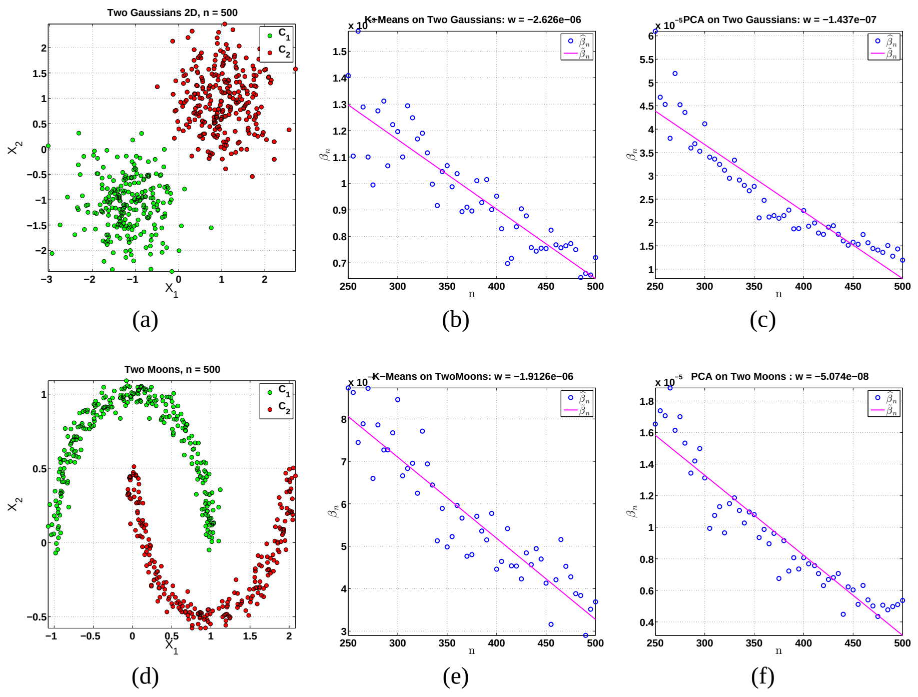


**Fig. 1. Left:** Two synthetic data sets, (a) two normally distributed clouds of points with equal variance and equal priors, and (d) two moons data points with equal priors. **Middle:** The estimated $\widehat{\beta}_n$ (blue circles) from Algorithm 1 for $k$–Means clustering on the two synthetic data sets. The fitted stability lines are shown in magenta. The slope of the stability lines is indicated by w. **Right:** The estimated $\widehat{\beta}_n$ and stability lines for PCA on the two synthetic data sets. The dispersion of $\widehat{\beta}_n$'s around the stability line is reflected in the norm of the residuals for the stability line (not displayed). Note the difference in the dispersion of points around the stability line for $k$–Means and PCA. Note also that the more structure in the two moons data set is reflected in a smaller w (compared to w for the tow Gaussians) for both algorithms.

the variance in $\{R_1, \ldots, R_m\}$ and $\{R'_1, \ldots, R'_m\}$. However, increasing $m$ beyond a certain value will not increase the accuracy of the estimated empirical loss. Reducing the variance in $\{R_1, \ldots, R_m\}$ and $\{R'_1, \ldots, R'_m\}$, in turn, encourages reducing the variance in $\{B_1, \ldots, B_m\}$. Note that for any random variable $Z$ with mean $\mu$, median $\nu$, and variance $\sigma^2$, then $|\mu - \nu| \leq \sigma$ with probability one. Therefore, in practice, increasing $m$ encourages reducing the variance in $B_j$'s thereby reducing the difference $|\widehat{\beta}_{n_t} - \mathbb{E}(B_j)|$. Observe that the operator $\max_{i=1,\ldots,n_t}$ defined $\forall \mathcal{S} \in \mathcal{X}^{n_t}$ in (6) is now replaced with the estimate $\widehat{\beta}_{n_t}$.

## 3.2 The Trend of $\widehat{\beta}_n$ and The Stability Line

The output of Algorithm 1 is the set $\mathcal{B}$ of estimated $\widehat{\beta}_{n_t}$'s for the increasing values of $n_t$. In order to assess the stability of A, we need to observe whether $\widehat{\beta}_{n_t} = \frac{\kappa}{n_t}$, for some constant $\kappa > 0$. As an example, Figure 1 shows the trend of $\widehat{\beta}_n$ for $k$–Means clustering and principal component analysis (PCA) on two synthetic toy data sets. The blue circles in the middle and right figures are the estimated $\widehat{\beta}_n$ from Algorithm 1.[5] Observe that $\widehat{\beta}_n$ is decreasing as $n$ is increasing.

To formally detect and quantify this decrease, a line is fitted to the estimated $\widehat{\beta}_n$ (shown in magenta); i.e. $\beta(n_t) = wn_t + \zeta$, where $w$ is the slope of the line, and $\zeta$ is the intercept. We call this line, the *Stability Line*. The slope of the stability line indicates its steepness which is an esimtate for the decreasing rate of $\beta_n$. For stable algorithms, $w < 0$, and $|w|$ indicates the stability degree of the algorithm. Note that $w = \tan\theta$, where $\theta$ is the angle between the stability line and the abscissa, and $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$. For $0 \leq \theta < \frac{\pi}{2}$, $\mathsf{A}$ is not stable. For $-\frac{\pi}{2} < \theta < 0$, if $\theta$ is approaching 0, then $\mathsf{A}$ is a less stable algorithm, while if $\theta$ is approaching $-\frac{\pi}{2}$, then $\mathsf{A}$ is a more stable algorithm. Observe that in this setting, $\beta$ is a function of $n$ and $w$, and hence it can be denoted by $\beta(n, w)$. Plugging $\beta(n, w)$ in the inequalities of Corollary 1, we get that:

$$(i) \quad R(\mathsf{A}_{\mathcal{S}}) \leq \widehat{R}_{\text{EMP}}(\mathsf{A}_{\mathcal{S}}) + 2(wn + \zeta) + [4n(wn + \zeta) + c]\sqrt{\frac{\log(1/\delta)}{2n}}, \tag{8}$$

$$(ii) \quad R(\mathsf{A}_{\mathcal{S}}) \leq \widehat{R}_{\text{LOO}}(\mathsf{A}_{\mathcal{S}}) + (wn + \zeta) + [4n(wn + \zeta) + c]\sqrt{\frac{\log(1/\delta)}{2n}}. \tag{9}$$

That is, the steeper is the stability line ($w < 0$), the more tight is the confidence bound. Figure 2 shows other examples for stability lines on the synthetic data sets (Gaussians and Moons) using Laplacian eigenmaps (LEM) [4], and Local Linear Embedding (LLE) [19]. The generalization assessment is based on the Laplacian matrix $\mathbf{L}$ for LEM, and the weighted affinity matrix $\mathbf{W}$ for LLE. In particular, the loss for LEM is $\ell = \mathrm{tr}(\mathbf{L}\mathbf{L}^\top)$, while for LLE, $\ell = \mathrm{tr}(\mathbf{W}\mathbf{W}^\top)$.

### 3.3 Comparing Two Algorithms: $\mathsf{A}_1$ vs. $\mathsf{A}_2$

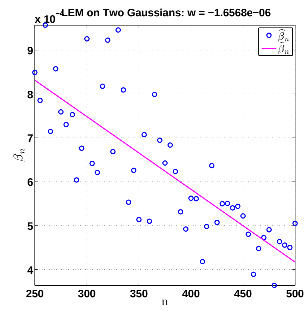
The previous generalization assessment procedure only considered one algorithm. Here we propose an extension for the above procedure to compare the generalization of two unsupervised learning algorithms, $\mathsf{A}_1$ and $\mathsf{A}_2$, under the same loss $\ell$, on a given data source. More specifically, the comparative setting addresses the following questions: if $\mathsf{A}_1$ is stable with respect to $\ell$ on $\mathcal{S}$ (according to Definition 2), and if $\mathsf{A}_2$ is stable with respect to $\ell$ on $\mathcal{S}$ (according to Definition 2), which algorithm has better generalization on $\mathcal{S}$? The following definition gives a formal answer to these questions.

**Definition 3 (Comparing $\mathsf{A}_1$ vs. $\mathsf{A}_2$ ).** *Let $\mathsf{A}_1$ be a stable algorithm with respect to $\ell$ on $\mathcal{S}$ with slope $w_1 < 0$ for its stability line. Let $\mathsf{A}_2$ be a stable algorithm with respect to $\ell$ on $\mathcal{S}$ with slope $w_2 < 0$ for its stability line. We say that:*

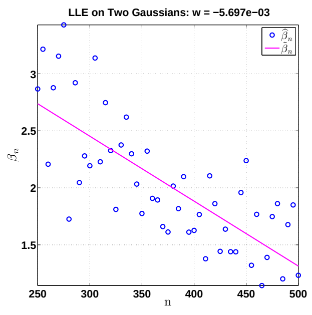
*1. $\mathsf{A}_1$ is similar to $\mathsf{A}_2$, denoted $\mathsf{A}_1 = \mathsf{A}_2$, if $w_1 \approx w_2$.*[6]

[5] In these experiments, $m = 100$, and $n_1, n_2, \ldots, n_\tau$ were set to $0.5n, 0.51n, \ldots, 0.99n, n$. The loss $\ell$ for $k$–Means is the sum of $L_1$ distances between each point and its nearest centre, and for PCA, $\ell = \mathrm{tr}(\mathbf{C})$, where $\mathbf{C}$ is the data's sample covariance matrix.

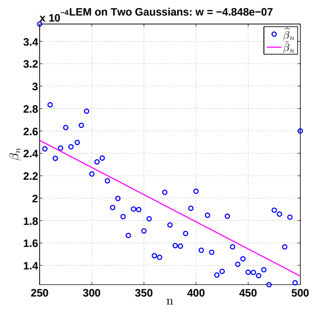
[6] This is done using hypothesis testing for the equality of slopes – See Appendix for details.



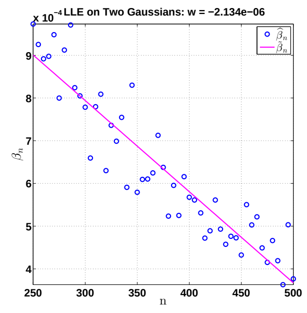
(a) LEM *nn* = 3



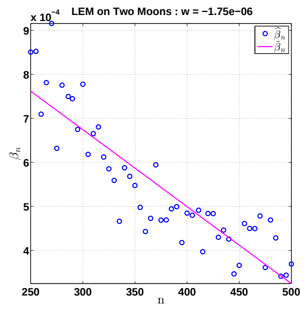
(b) LLE *nn* = 3



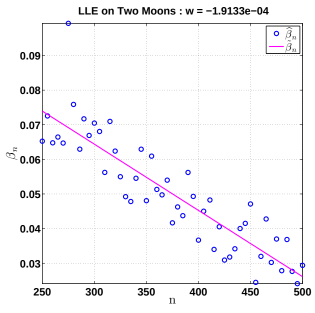
(c) LEM *nn* = 10



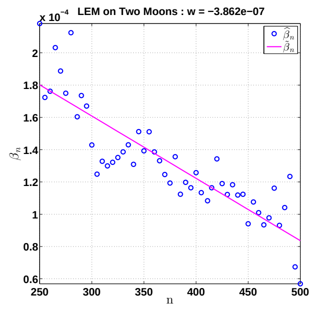
(d) LLE *nn* = 15



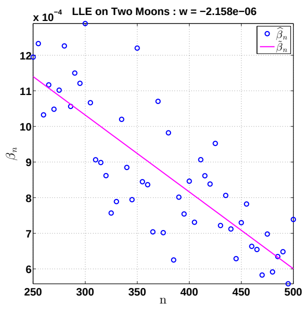
(e) LEM *nn* = 3



(f) LLE *nn* = 5

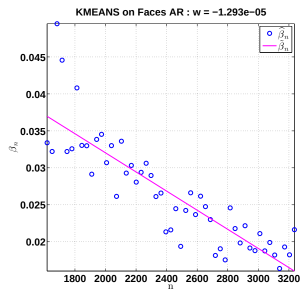


(g) LEM *nn* = 10

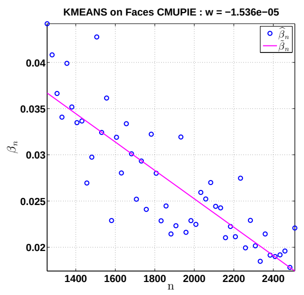


(h) LLE *nn* = 15

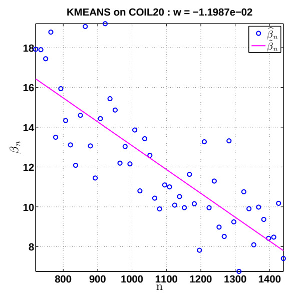
**Fig. 2. First Column:** Generalization assessment for LEM on two Gaussians (a,c) and two moons (e,g), with different number of nearest neighbours (*nn*) for constructing the data's neighbourhood graph. Compare the slopes (`w`) for the stability lines and the dispersion of points around it, and note the sensitivity of LEM to the number of *nn*. The same follows for the two moons case (e,g). Note also the difference in the stability lines (slope, and dispersion of estimated $\widehat{\beta}_n$'s) for LEM and PCA on the same data sets. **Second Column:** Generalization assessment for LLE on two Gaussians (b,d) and two moons (f,h) data sets, with different number of *nn*.



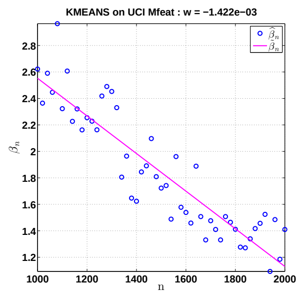
(a) Faces AR



(b) Faces CMUPIE



(c) Coil20



(d) UCI MFeat

**Fig. 3.** Generalization assessment for $k$–Means clustering using stability analysis on four real data sets: (a) Faces AR, (b) Faces CMUPIE, (c) Coil20, and (d) UCI MFeat.

2. $\mathsf{A}_1$ *is better than* $\mathsf{A}_2$*, denoted by* $\mathsf{A}_1 \succ \mathsf{A}_2$*, if* $w_1 < w_2$.
3. $\mathsf{A}_1$ *is worse than* $\mathsf{A}_2$*, denoted* $\mathsf{A}_1 \prec \mathsf{A}_2$*, if* $w_1 > w_2$.

To develop a formal procedure for such an assessment we proceed by letting Algorithm 1 invoke the two algorithms $\mathsf{A}_1$ and $\mathsf{A}_2$ on the same subsamples $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ and $\{\mathbf{X}'_1, \ldots, \mathbf{X}'_m\}$. The final output of Algorithm 1 will be two sets $\mathcal{B}^1 = \{\widehat{\beta}^1_{n_1}, \ldots, \widehat{\beta}^1_{n_\tau}\}$, and $\mathcal{B}^2 = \{\widehat{\beta}^2_{n_1}, \ldots, \widehat{\beta}^2_{n_\tau}\}$. The analysis then proceeds by fitting the stability line for each algorithm, plotting the curves shown in Figures 1 and 2, and then comparing the slopes of both algorithms. Formal comparison for the slopes $w_1$ and $w_2$ is done using hypothesis testing for the equality of slopes:

$$H_0 : w_1 = w_2 \quad \text{vs.} \quad H_1 : w_1 \neq w_2.$$

If $H_0$ is rejected at a significance level $\alpha$ (usually $0.05$ or $0.01$), then deciding which algorithm has better generalization can be done using rules 2 and 3 in the above definition. If $H_0$ cannot be rejected at the desired significance level, then both algorithms have a similar generalization capability. Further insight can be gained through the norm of the residuals, and the spread of the estimated $\widehat{\beta}_n$'s around the stability line.

## 4 Empirical Validation on Real Data Sets

We have conducted some initial validation tests for the proposed generalization assessment framework. In these experiments, we considered two different unsupervised learning problems: clustering and dimensionality reduction (linear and nonlinear). In particular, we considered the following algorithms: $k$–Means for clustering, PCA for linear dimensionality reduction, and LEM and LLE for nonlinear dimensionality reduction (NLDR). The four algorithms were run on four data sets from different domains: (*i*) two faces data sets, AR and CMUPIE with (samples $\times$ features) $3236 \times 2900$, and $2509 \times 2900$, respectively. (*ii*) two image features data sets, Coil20 and Multiple Features (MFeat) with (samples $\times$ features) $1440 \times 1024$, and $2000 \times 649$, respectively, from the UCI Repository for Machine Learning [18]. [7] In all these experiments, the number of bootstraps $m$ was set to $100$, and the values for $n_1, n_2, \ldots, n_\tau$ were set to $0.5n, 0.51n, 0.52n, \ldots, 0.99n, n$, where $n$ is the original size of the data set.

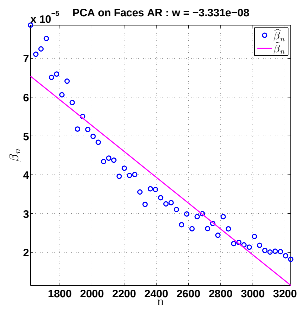
To apply the proposed generalization assessment, an external loss $\ell$ needs to be defined for each problem. $k$–Means minimizes the sum of $L_2$ distances between each point and its nearest cluster centre. Thus, a suitable loss can be the sum of $L_1$ distances. Note that the number of clusters $k$ is assumed to be known. Note also that in this setting, for each iteration $j$ in Algorithm 1, the initial $k$ centres are randomly chosen and they remain unchanged after holding out the random sample. That is, $k$–Means starts from the same initial centres before and after holding out one sample.

For PCA, LEM and LLE, the loss functions are chosen as follows: $\ell = \mathrm{tr}(\mathbf{C})$ for PCA, $\ell = \mathrm{tr}(\mathbf{L}\mathbf{L}^\top)$ for LEM, and $\ell = \mathrm{tr}(\mathbf{W}\mathbf{W}^\top)$ for LLE, where $\mathbf{C}$ is the data's sample covariance matrix, $\mathbf{L}$ is the Laplacian matrix defined by LEM, and $\mathbf{W}$ is the weighted affinity matrix defined by LLE. The number of nearest neighbours for constructing the neighbourhood graph for LEM and LLE was fixed to $30$ to ensure that the neighbourhood graph is connected. Note that we did not perform any model selection for the number of nearest neighbours to simplify the experiments and the demonstrations.

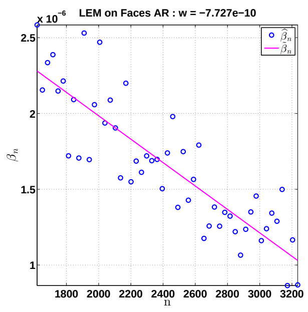
## 4.1 Generalization Assessment of $k$–Means Clustering

Figure 3 shows the stability lines for $k$–Means clustering on the four real data sets used in our experiments. For both faces data sets, AR and CMUPIE, the stability lines have similar slopes despite the different sample size. However, the dispersion of points around the stability line is bigger for CMUPIE than it is for AR. Hypothesis testing for the equality of slopes (at significance level $\alpha = 0.05$) did not reject $H_0$ ($p$–value = 0.92). For Coil20 and UCI Mfeat, the slopes of stability lines differ by one order of magnitude (despite the different sample size). Indeed, the hypothesis test in this case rejected $H_0$ with a very small $p$–value. Note that the estimated $\widehat{\beta}_n$'s for the four data sets do not show a clear trend as is the case for the two Gaussians and the two moons data sets in Figure 1. This behaviour is expected from $k$–Means on real high dimensional data sets, and is in agreement with what is known about its sensitivity to the initial centres and its convergence to local minima. For a better comparison, observe the stability lines for PCA on the same data sets in Figures 4 and 5.

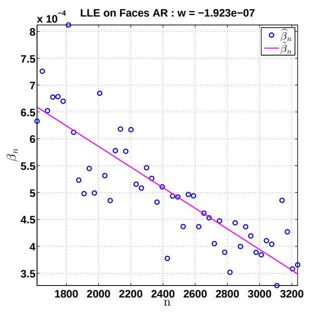
[7] The AR and CMUPIE face data sets were obtained from `http://www.face-rec.org/databases/`.



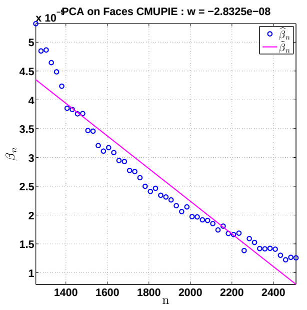
(a) PCA on Faces AR



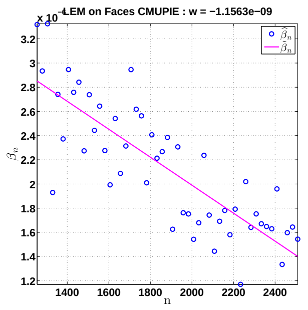
(b) LEM on Faces AR



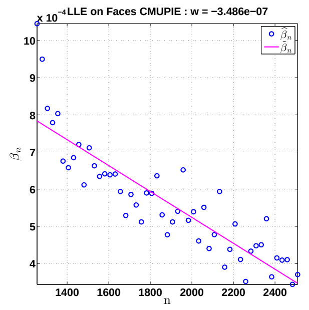
(c) LLE on Faces AR



(d) PCA on Faces CMUPIE



(e) LEM on Faces CMUPIE



(f) LLE on Faces CMUPIE

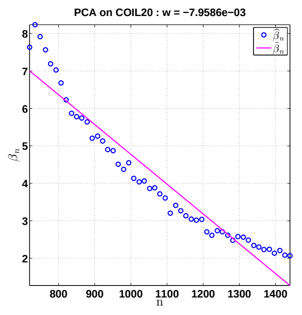
**Fig. 4.** Generalization assessment for PCA, LEM and LLE using stability analysis on two faces data sets: AR (a,b,c), and CMUPIE (d,e,f).

## 4.2 Generalization Assessment of PCA, LEM, and LLE

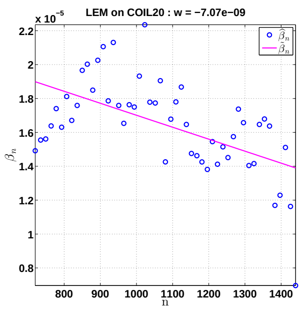
Figures 4 and 5 show the stability lines for the three dimensionality reduction algorithms; PCA, LEM and LLE, on the four real data sets used in our experiments. Note that the magnitude of $w$ for these experiments should not be surprising given the scale of $n$ and $\widehat{\beta}_{n_t}$. It can be seen that PCA shows a better trend of the estimated $\widehat{\beta}_n$'s than LEM and LLE (for our choice of fixed neighbourhood size). This trend shows that PCA has better stability (and hence better generalization) than LEM and LLE on these data sets. Note that in this setting, the slope for PCA stability line cannot be compared to that of LEM (nor LLE) since the loss functions are different. However, we can compare the slopes for each algorithm stability lines (separately) on the face data sets and on the features data sets.

Hypothesis testing ($\alpha = 0.05$) for PCA stability lines on AR and CMUPIE rejects $H_0$ in favour of $H_1$ with a $p$–value $= 0.0124$. For Coil20 and UCI Mfeat, the test did not reject $H_0$ and the $p$–value $= 0.9$. For LEM, the test did not reject $H_0$ for the slopes of AR and CMUPIE, while it did reject $H_0$ in favour of $H_1$ for Coil20 and UCI MFeat. A similar behaviour was observed for LLE.

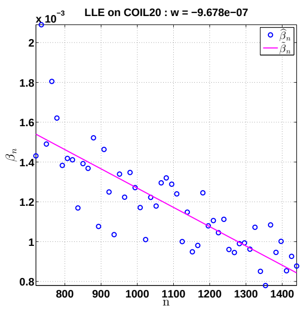
In these experiments and the previous ones on $k$–Means clustering, note that no comparison of two algorithms were carried on the same data set. In these illustrative examples, the generalization of one algorithm was assessed on two different data sets,



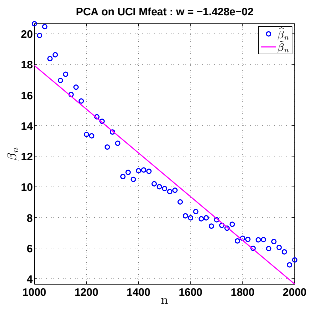
(a) PCA on Coil20



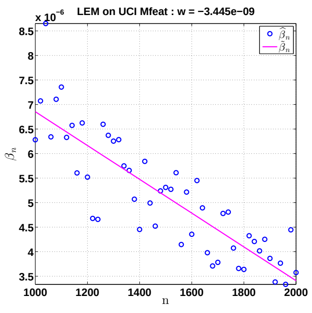
(b) LEM on Coil20



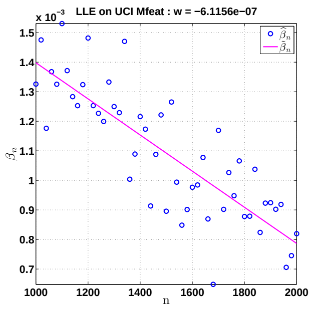
(c) LLE on Coil20



(d) PCA on MFeat



(e) LEM on MFeat



(f) LLE on MFeat

**Fig. 5.** Generalization assessment for PCA, LEM and LLE using stability analysis on Coil20 (a,b,c), and UCI MFeat (d,e,f).

following the examples on the synthetic data sets in Figures 1. Note that this scenario is different from the one described in § 3.3. In the above experiments, the trend of $\widehat{\beta}_{n_t}$, the stability line, the slope $w$, and the scatter of points around the stability line, provided a quantitative and a qualitative evaluation for the generalization capability of $k$–Means and PCA. However, our experience suggests that when analyzing the generalization of one algorithm on two different data sets, hypothesis testing can give more accurate insight if the sample sizes $n_t$ are equal for both data sets since $\beta_n$ is known to decrease as $\kappa/n$, and $\kappa > 0$.

## 5 Concluding Remarks

In this paper we proposed a general criterion for generalization in unsupervised learning that is analogous to the prediction error in supervised learning. We also proposed a computationally intensive, yet efficient procedure to realize this criterion on finite data sets, and extended it for comparing two different algorithms on a common data source. Our preliminary experiments showed that, for algorithms from three different unsupervised learning problems, the proposed framework provided a unified mechanism and a unified interface to assess their generalization capability. This type of analysis suggests further rigorous assessment of these algorithms, and is encouraging to conduct similar analysis for other unsupervised learning problems such as density estimation, subspace clustering, feature learning, and layer wise analysis of deep architectures. Further, our framework can be extended to answer model selection questions for unsupervised learning, or it can be complimentary to exiting methods for model selection.

## Acknowledgments

We would like to thank our Reviewers for their helpful comments and suggestions, the Alberta Innovates Centre for Machine Learning and NSERC for their support, and Frank Ferrie for additional computational support at McGill's Centre for Intelligent Machines.

## Appendix

Hypothesis testing for the equality of slopes $w_1$ and $w_2$ for two regression lines $Y_1 = w_1 X_1 + \zeta_1$ and $Y_2 = w_2 X_2 + \zeta_2$, respectively, proceeds as follows. Let $\mathcal{S}_1 = \{(x_1^i, y_1^i)\}_{i=1}^{n_1}$ and $\mathcal{S}_2 = \{(x_2^j, y_2^j)\}_{j=1}^{n_2}$, be the two data sets to be used for estimating the lines defined by $w_1$ and $w_2$, respectively. Let $\{\widehat{y}_1^1, \ldots, \widehat{y}_1^{n_1}\}$ and $\{\widehat{y}_2^1, \ldots, \widehat{y}_2^{n_2}\}$ be the estimated predictions from each regression line. The null and alternative hypotheses of the test are:

$$H_0 : w_1 = w_2 \quad \text{vs.} \quad H_1 : w_1 \neq w_2.$$

That is, $H_0 : w_1 - w_2 = 0$. If $H_0$ is true, then $w_1 - w_2 \sim \mathscr{G}(0, s_{w_1 w_2})$, where $s_{w_1 w_2}$ is the pooled error variance. Using a $t$ test, we construct the statistic $t$:

$$t = \frac{w_1 - w_2}{s_{w_1 w_2}} \sim \mathscr{T}_r \, ,$$

where $\mathscr{T}_r$ is the Student's $t$ distribution with $r$ degrees of freedom, and $r = n_1 + n_2 - 4$. The pooled error variance is defined as:

$$s_{w_1 w_2} = \sqrt{s_{w_1}^2 + s_{w_2}^2},$$

where

$$s_{w_k}^2 = \frac{e_k}{\sigma_k^2 (n_k - 1)},$$

$e_k = \sum_{i=1}^{n_k} (y_k^i - \widehat{y}_k^i)^2 / (n_k - 2)$, and $\sigma_k^2 = Var(X_k)$, which can be replaced by the sample variance. For significance level $\alpha$, we compute the probability of observing the statistic $t$ from $\mathscr{T}_r$ given that $H_0$ is true; this is the $P$ value of the test. If $P > \alpha$, then $H_0$ cannot be rejected. Otherwise, reject $H_0$ in favour of $H_1$.

## References

1. Anandkumar, A., Hsu, D., Kakade, S.: A method of moments for mixture models and hidden Markov models. CoRR abs/1203.0683 (2012)

2. Balle, B., Quattoni, A., Carreras, X.: Local loss optimization in operator models: A new insight into spectral learning. In: Proceedings of the 29th International Conference on Machine Learning. pp. 1879–1886 (2012)
3. Bartlett, P., Mendelson, S.: Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research 3, 463–482 (2003)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for data representation. Neural Computation 15, 1373–1396 (2003)
5. Bousquet, O., Elisseeff, A.: Stability and generalization. Journal of Machine Learning Research 2, 499–526 (2002)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
7. Devroye, L., Györfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Springer, New York (1996)
8. Devroye, L., Wagner, T.: Distribution-free inequalities for the deleted and holdout error estimates. IEEE Transactions on Information Theory 25(2), 202–207 (1979)
9. Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
10. Efron, B.: Bootstrap methods: another look at the jackknife. Annals of Statistics 7, 1–26 (1979)
11. Hansen, L., Larsen, J.: Unsupervised learning and generalization. In: Proceedings of the IEEE International Conference on Neural Networks. pp. 25–30 (1996)
12. Hsu, D., Kakade, S., Zhang, T.: A spectral algorithm for learning hidden markov models. In: Proceedings of the 22nd Conference on Learning Theory (2009)
13. Jordan, M.: On statistics, computation and scalability. Bernoulli 19(4), 1378–1390 (2013)
14. Kearns, M., Ron, D.: Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In: Proceedings of the Conference on Learning Theory. pp. 152–162. ACM (1999)
15. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. pp. 1137–1143 (1995)
16. Kutin, S., Niyogi, P.: Almost-everywhere algorithmic stability and generalization error. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence. pp. 275–282 (2002)
17. Mukherjee, S., Niyogi, P., Poggio, T., Rifkin, R.: Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. Advances in Computational Mathemathics 25, 161–193 (2006)
18. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI Repository of Machine Learning Databases (1998), `www.ics.uci.edu/∼mlearn/MLRepository.html`
19. Saul, L., Roweis, S.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4, 119–155 (2003)
20. Shalev-Shwartz, S., Shamir, O., Srebro, N., Sridharan, K.: Learnability, stability and uniform convergence. Journal of Machine Learning Research 11, 2635–2670 (2010)
21. Song, L., Boots, B., Siddiqi, S., Gordon, G., Smola, A.: Hilbert space embeddings of hidden Markov models. In: Proceedings of the 27th International Conference on Machine Learning. pp. 991–998 (2010)
22. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Sussex, England (1998)
23. Vapnik, V.N.: An overview of statistical learning theory. IEEE Transactions on Neural Networks 10(5), 988–999 (Sep 1999)
24. Xu, H., Mannor, S.: Robustness and generalization. Machine Learning 86(3), 391–423 (2012)
25. Xu, L., White, M., Schuurmans, D.: Optimal reverse prediction: a unified perspective on supervised, unsupervised and semi-supervised learning. In: Proceedings of the International Conference on Machine Learning. vol. 382 (2009)