


Bootstrap Confidence Regions for Learned Feature Embeddings

Kris Sankaran


To cite this article: Kris Sankaran (2023): Bootstrap Confidence Regions for Learned Feature Embeddings, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2023.2197478


To link to this article: <https://doi.org/10.1080/10618600.2023.2197478>

 View supplementary material [↗](#)

 Published online: 15 May 2023.

 Submit your article to this journal [↗](#)

 Article views: 26

 View related articles [↗](#)

 View Crossmark data [↗](#)



Bootstrap Confidence Regions for Learned Feature Embeddings

Kris Sankaran^{a,b}

^aDepartment of Statistics, University of Wisconsin - Madison, Madison, WI; ^bWisconsin Institute for Discovery, Madison, WI

ABSTRACT

Algorithmic feature learners provide high-dimensional vector representations for non-matrix structured data, like image or text collections. Low-dimensional projections derived from these representations, called embeddings, are often used to explore variation in these data. However, it is not clear how to assess the embedding uncertainty. We adapt methods developed for bootstrapping principal components analysis to the setting where features are algorithmically derived from nonmatrix data. We empirically compare the derived confidence areas in simulations, varying factors influencing feature learning and the bootstrap, like feature learning algorithm complexity and bootstrap sample size. We illustrate the proposed approaches on a spatial proteomics dataset, where we observe that embedding precision is not uniform across all tissue types. Code, data, and pretrained models are available as an R compendium in the supplementary materials. Supplementary files for this article are available online.

ARTICLE HISTORY

Received February 2021
Accepted March 2023

KEYWORDS

Bootstrap/resampling; Data visualization; Dimension reduction; Machine learning

Algorithmic feature learning has transformed modern data analysis, enabling the exploration and manipulation of complex data types through simple vector representations. In this way, data as diverse as molecular graphs, satellite images, and document collections have been made amenable to exploration using classical multivariate analysis tools. In particular, when these learned representations are high-dimensional, it is natural to apply dimensionality reduction to them, both to support qualitative interpretation and as a denoising step before subsequent modeling (Erhan et al. 2009; Nguyen, Yosinski, and Clune 2019). These dimensionality-reduced representations, also called embeddings, are widely used; however, it remains a challenge to quantify the uncertainty associated with the resulting projections. This uncertainty can be beneficial to both interpretation and modeling. For example, the vector representation of a short document may be more ambiguous than that of a long one simply because fewer words are available to judge its content. Treating the derived projections of the two documents as equally precise would not be appropriate. Indeed, if we assume that the documents were drawn from a topic model, then the number of words in each document equals the number of trials in a multinomial sample centered around the unknown representation (Ke and Wang 2022).

To characterize embedding uncertainty in algorithmic feature learners, we adapt bootstrap methods for building confidence areas for principal components analysis (Elguero and Holmes-Junca 1988; Chateau and Lebart 1996; Josse, Wager, and Husson 2016). These methods cannot be directly applied to embeddings derived from projecting learned features because such an analysis fails to account for randomness in the feature learning process, resulting in underestimated uncertainty. Indeed, it is known that treating the output of a preliminary

model as fixed can bias inferences in downstream analysis, and developing valid inference procedures in this more complex setting is an area of active interest. For example, Kim et al. (2020) provide a strategy for valid post-dimension reduction inference in the sufficient dimensionality reduction setting, and Wang, McCormick, and Leek (2020) proposes a sample-splitting method for valid post-prediction inference. For concreteness, in word embedding applications, training an embedding model twice may result in two different, though hopefully similar, representations. This introduces randomness in the feature extraction process, and the coordinates of the two sets of learned representations need not correspond to one another, in contrast to the usual setting where the columns of the data matrix are held constant. Assuming that the output from the word embedding algorithm is fixed could lead to underestimated uncertainty during downstream modeling or visualization, and improved measures of embedding variability could help calibrate subsequent inferences. A final source of complexity is that many feature learners are supervised, and if the model is overfit, the associated embeddings may suggest misleading interpretations, overstating the differences between classes in the learned feature space (Gross, Taylor, and Tibshirani 2015).

To characterize the precision of embeddings derived from feature learners, we propose to build confidence areas for these embeddings by bootstrapping the entire workflow, including retraining the original feature extraction algorithms. Specifically, we compare the computational efficiency and statistical coverage for three bootstrap strategies motivated by variations of the nonparametric and parametric bootstrap for PCA. Though all bootstrap approaches give comparable results and are approximately valid in a simple low-rank model, more complex simulations suggest that the parametric bootstrap underestimates

embedding uncertainty across all feature extractors. Therefore, multiple runs of the feature learner appear necessary for uncertainty estimation. Our experiments also identify that the choice of feature extractor can strongly affect embedding variability, but model complexity and bootstrap hyperparameters have comparatively little influence. Through an application to a spatial proteomics dataset, we illustrate how confidence areas can highlight qualitative distinctions between regions of the embedding space. We have released all code, raw and intermediate data, and trained models associated with simulations and data analysis. Documentation about these artifacts and how to reproduce them is provided in Supplementary Section 1 and through the research compendium at <https://github.com/krisrs1128/LFBCR>.

1. Background

1.1. Feature Learning

We will analyze projections from three complementary feature learning algorithms. These techniques are common examples of deep unsupervised, deep supervised, and shallow feature learning algorithms used in practice. The first algorithm is the Variational Auto-Encoder (VAE), an unsupervised method that learns a K -dimensional reduction of a dataset by optimizing a reconstruction objective (Kingma and Welling 2014). It models samples x using the generative mechanism $p(z)p_\xi(x|z)$ of the data; $p(z)$ is a prior on latent features and $p_\xi(x|z)$ is a likelihood parameterized by ξ . The algorithm finds a pair ξ, φ maximizing the lower bound,

$$\log p(x) \geq \mathbb{E}_{q_\varphi} [\log p_\xi(x|z)] - D_{\text{KL}}(q_\varphi(z|x)||p(z))$$

where D_{KL} is the Kullback-Leibler divergence between two probability distributions and $q_\varphi(z|x) = \mathcal{N}(\mu_\varphi(x), \text{diag}(\sigma_\varphi^2(x)))$ maps samples to normal distributions in the latent space with sample-dependent means $\mu_\varphi(x)$ and variances $\sigma_\varphi^2(x)$. This problem is nonconvex, and the solution is nondeterministic. There are many implementations of VAEs, and the experiments below apply the interface released by Van Den Oord and Vinyals (2017).

Second, we learn supervised features through a Convolutional Neural Network (CNN). Unlike the VAE, this model has access to responses $y \in \mathbb{R}$ for each sample and attempts to learn representations that predict y . A CNN regressor optimizes an empirical estimate of $\mathbb{E}\|y - f_{W_{1:j}}(x)^T \beta\|_2^2$ over $W_{1:j}$ and β . Here, $f_{W_{1:j}}$ passes sample x through J layers, each of which learns more complex features associated with the input before arriving at a ‘‘final layer’’ representation used for predictions. The final representation $f^J(x)$ is defined recursively according to

$$\begin{aligned} f_{W_{1:j}}^j(x) &= \sigma\left(W_j f_{W_{1:(j-1)}}^{j-1}(x)\right) \\ f^0(x) &= x \end{aligned}$$

where $\sigma(x) := x\mathbb{I}(x \geq 0)$ is a rectified linear unit and matrices W_j are restricted to the set of convolutions. Like in the VAE, this is solved through first-order optimization methods. Our implementation follows the Convolution-Batch Normalization-Rectified Linear Unit architecture from Raghu et al. (2017).

Third, we use a random convolutional features (RCF) model (Rahimi and Recht 2008), a shallow alternative to the deep feature extractors above. A random sample of K training examples $x_{i_1}, \dots, x_{i_K} \in \mathbb{R}^{w \times h \times c}$ is selected; the x_i 's are assumed to be c -channel images with dimension $w \times h$. A random $s \times s$ patch, denoted $w_k \in \mathbb{R}^{s \times s \times c}$, is extracted for each sample. For any c -channel image x , the k th feature z_k is found by convolving x with w_k and spatially averaging over activations. This model uses random training image patches as convolutional kernels rather than learning them from scratch. The concatenated features $[z_1, \dots, z_K]$ are analogous to the features $f_{W_{1:j}}(x)$ in the CNN. For prediction, the n_{train} training samples are featureized into $\mathbf{Z} \in \mathbb{R}^{n_{\text{train}} \times K}$. Then, a ridge regression model is trained from \mathbf{Z} to the y , giving an estimate $\hat{\beta}$. For a new example x^* , the same image patches w_1, \dots, w_K are used to form z^* , and predictions are made with $z^{*T} \hat{\beta}$. This model does not require gradient-based training and can be learned quickly. Nonetheless, its dependence on the random selection x_{i_k} means that results are not guaranteed to agree from run to run.

1.2. Principal Components Analysis and the Bootstrap

Several methods are available for bootstrapping Principal Components Analysis (PCA), and these provide the foundation for analyzing embedding uncertainty in feature learning algorithms. The total bootstrap computes B principal planes by applying PCA to B resampled versions of the data (Elguero and Holmes-Junca 1988; Chateau and Lebart 1996). For each b , rows are sampled with replacement, viewed as draws from a larger population. If \mathbf{X}_b is the b th resampled version of the original data $\mathbf{X} \in \mathbb{R}^{n \times D}$, and if $\mathbf{X}_b = \mathbf{U}_b \mathbf{\Sigma}_b \mathbf{V}_b^T$ is its associated rank- K truncated Singular Value Decomposition (SVD), then each plane is the span of $\mathbf{V}_b \in \mathbb{R}^{p \times K}$. The associated principal axes may be reflected or swapped with one another, so the associated sample coordinates must be aligned. Procrustes analysis can support this alignment. Specifically, \mathbf{V}^b can be used to derive projected coordinates $\mathbf{Z}^b = \mathbf{X} \mathbf{V}^b$ for the original data. Procrustes analysis identifies rotation matrices $\mathbf{R}_1, \dots, \mathbf{R}_B$ and a mean matrix \mathbf{M} minimizing,

$$\min_{\mathbf{R}_1, \dots, \mathbf{R}_B \in \mathcal{O}(K, K), \mathbf{M}} \sum_{b=1}^B \|\mathbf{Z}_b \mathbf{R}_b - \mathbf{M}\|_F^2,$$

where $\mathcal{O}(K, K)$ is the space of $p \times p$ orthonormal matrices. This problem can be solved by an algorithm that cyclically optimizes each \mathbf{R}_b and \mathbf{M} in turn (Friedman, Hastie, and Tibshirani 2001). It returns a cloud of B points $\mathbf{R}_1 z_i, \dots, \mathbf{R}_B z_i$ for each sample i , and their sample mean and covariance can be used to derive a level- α confidence ellipsoid.

In contrast, fixed-effects PCA views the rows of the data matrix as the entire population of interest (Josse, Wager, and Husson 2016). The source of randomness, in this case, is measurement noise around a low-rank model, not sampling from a larger population of rows. A parametric bootstrap provides confidence areas for the true latent coordinates in a low-rank model by estimating the model and resampling residuals. We will provide relevant details of this approach when we adapt it to the feature learning setting in Section 2.2.

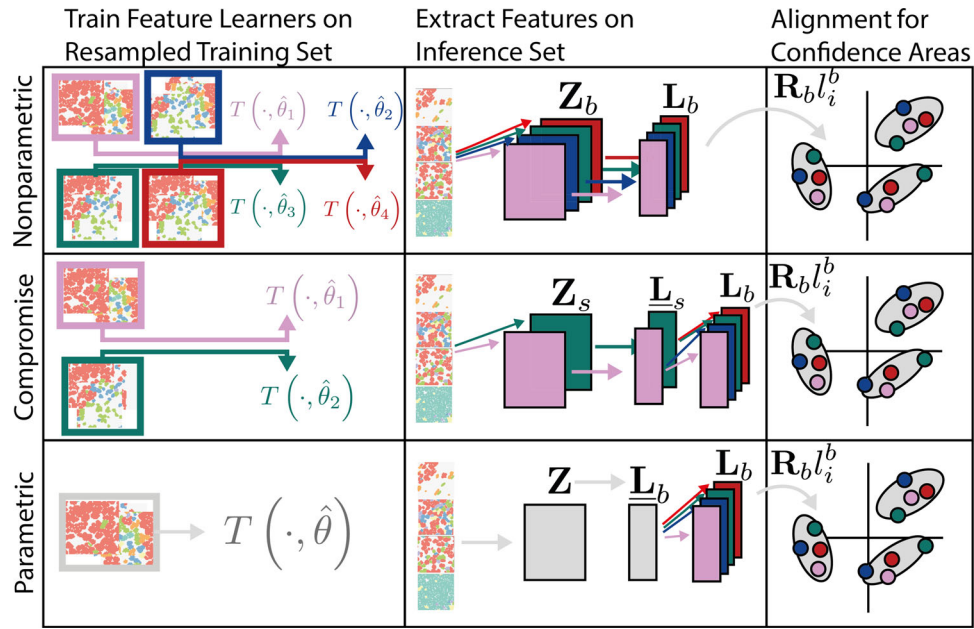


Figure 1. A summary of the proposed bootstrap procedures. All begin by splitting data into training and inference sets for feature learning and embedding generation. The nonparametric bootstrap (top row) trains B separate feature learners, and each is used for feature extraction and dimensionality reduction before being aligned. The parametric bootstrap (bottom row) trains a single feature learner and then simulates and aligns an ensemble of B latent coordinates for each sample. The compromise (middle row) trains a smaller set of feature learners but further resamples residuals to increase the number of apparent bootstrap replicates.

2. Bootstrap Strategies

This section adapts PCA-based bootstrap methods to the feature learning setting. Figure 1 summarizes our three candidate approaches. These bootstrap methods quantify the stability of the principal subspaces associated with the representations derived by feature learning algorithms. Two proposals are direct analogs of the nonparametric and parametric bootstrap for PCA. A challenge with using the nonparametric bootstrap directly is that it can be computationally expensive, requiring the training of many feature learners. In contrast, the parametric bootstrap only requires training a single feature learner. However, it makes stronger assumptions about the relationship between embeddings learned across training runs. Our third approach, given in Section 2.3, attempts to bridge these two extremes.

Before introducing the bootstraps in detail, we establish notation. Suppose we have n samples $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}$, where \mathcal{X} is the domain of the covariates x_i and y_i are labels. Collect all samples into $\mathcal{D} = (x_i, y_i)_{i=1}^n$. A *feature learner* is a parameterized mapping $T(\cdot; \theta) : \mathcal{X} \rightarrow \mathbb{R}^P$ taking data from \mathcal{X} and representing it in \mathbb{R}^P . For example, in a text data application, we expect the learner to transform a document into a vector of features reflecting its topic. θ is estimated from data by optimizing,

$$\hat{\theta} := \arg \min_{\theta \in \Theta} \mathcal{L}(\mathcal{D}, T(\cdot; \theta)) \quad (1)$$

for some loss \mathcal{L} . For an unsupervised feature learner, candidates $\theta \in \Theta$ are functions of x_i alone. For a supervised feature learner, the class includes functions of both x_i and y_i . To simplify notation, we will write $z_i = T(x_i; \hat{\theta}) \in \mathbb{R}^P$ to denote the learned features for observation i . A challenge is that the learned features are not the same from one run to the next—the k th learned feature from one run need not have any relationship with the k th feature from the next. This misalignment is a consequence

of using stochastic optimization in (1). However, even without direct correspondence across runs, they may all reflect the same underlying latent features. In particular, projections of learned features onto their principal subspaces may be similar across multiple runs after applying an appropriate alignment.

We will split the n samples into a feature learning set, indexed by $I \subset \{1, \dots, n\}$, and an inference set, indexed by I^C . This split is motivated by the potential to overfit features to a supervised learning task (Gross, Taylor, and Tibshirani 2015; Wang, McCormick, and Leek 2020). In this case, when analyzing the resulting embeddings, a strong gradient across values of y may appear, even if there is no relationship between the x_i and y_i , and using a separate inference set guards against this potentially misleading visualization. The setting is analogous to that encountered in post-selection inference—we do not select features based on their association with y , but we do algorithmically learn features that are. The fraction $\frac{1}{n}|I|$ used for feature learning is a hyperparameter whose influence is empirically studied below. The learning set $(x_i)_{i \in I}$ is resampled B times; in contrast, samples within the inference set I^C are fixed. The B resampled learning datasets are used to train B feature extractors, $T(\cdot; \hat{\theta}^b)$ which can then be applied to the entire dataset, yielding learned features $\mathbf{Z}_b \in \mathbb{R}^{n \times P}$.

2.1. Nonparametric Bootstrap

Like the total bootstrap, one approach to comparing embeddings across feature extractors is to learn B feature extractors on the training set, use them to extract projected features on the inference set, and then align the results. We implement this through the following process. For each b , compute the rank- K truncated singular value decomposition $\hat{\mathbf{Z}}_{b, I^C} = \mathbf{U}_b \mathbf{\Sigma}_b \mathbf{V}_b^T$, where the index I^C means that only features associated with the inference

set are used. We then define coordinates for sample i with respect to \mathbf{V}_b using the i th row of $\mathbf{L}_b := \mathbf{U}_b \boldsymbol{\Sigma}_b \in \mathbb{R}^{|I^C| \times K}$, which we refer to as the embedding l_i^b . These coordinates provide a low-dimensional representation of the learned features from the b th feature extractor.

A Procrustes analysis applied to $\mathbf{L}_1, \dots, \mathbf{L}_B$ learns a series of rotation matrices $\mathbf{R}_1, \dots, \mathbf{R}_B$ aligning the embeddings across bootstrap replicates. For each sample i , compute a sample mean and covariance matrix based on the B vectors $\mathbf{R}_b l_i^b$. These can then be used to create $1 - \alpha$ level confidence areas for each inference sample in the K -dimensional space of embeddings. Specifically, each sample's confidence area is an ellipse centered at the sample mean of the $\mathbf{R}_b l_i^b$ and with axes given by the eigenvectors of the associated sample covariance. This approach plugs in an estimate for a "true" low-dimensional \mathbf{L} , assuming that this representation is noisily observed and then subject to arbitrary rotations. Note that if the underlying latent representations are subject to more general transformations (e.g., translations) across runs of the extractor, this assumption would not be appropriate, a point that we revisit in Section 5.

The advantage of this approach is that it does not require a parametric model for simulating new versions of \mathbf{L}_b . The price to pay is that it is necessary to train B feature extraction models $T(\cdot, \hat{\theta}_b)$, which can be a computational burden, even if it is parallelizable. Further, confidence areas are not computed for samples in the feature learning set $(x_i)_{i \in I}$. However, suppose the uncertainty of sample-level projections is assumed to vary smoothly. Then, a heuristic is to consider the uncertainty of a training sample in I as comparable to those of nearby samples in the inference set I^C .

2.2. Parametric Bootstrap

We consider a parametric bootstrap for a low-rank model to avoid the computational complexity associated with training B feature extractors. Our approach follows the construction of confidence areas in fixed-effects PCA (Josse, Wager, and Husson 2016), except that we add a permutation at the final step, reflecting that the coordinates of the feature extractor need not match across runs. Specifically, this bootstrap simulates \mathbf{L}_b by resampling residuals from a fitted low-rank model, analogous to fixed-effects PCA. Suppose that variation across x_i is induced by latent features $l_i \in \mathbb{R}^K$. We model the feature learning process by,

$$\mathbf{Z} = \mathbf{L}\mathbf{V}^T + \mathbf{E} \quad E_{ij} \sim \mathcal{N}(0, \sigma_{\mathbf{E}}^2) \quad (2)$$

$$y = \mathbf{L}\beta + \epsilon \quad \epsilon_i \sim \mathcal{N}(0, \sigma_{\epsilon}^2) \quad (3)$$

where $\mathbf{L} \in \mathbb{R}^{n \times K}$ stacks the l_i and E_{ij} is the ij th element of \mathbf{E} . Only \mathbf{Z} is available for predicting the response y .

To simulate embeddings \mathbf{L}_b based on a single set of learned features \mathbf{Z} , we resample the rows of \mathbf{Z} in the inference set I^C , resulting in \mathbf{Z}_{b,I^C} . We compute the rank- K truncated SVD $\hat{\mathbf{Z}}_{b,I^C} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ of \mathbf{Z}_{b,I^C} . Letting $\underline{\mathbf{L}}_b = \mathbf{U}_b \boldsymbol{\Sigma}_b$, we then simulate bootstrap samples,

$$\mathbf{L}_b = (\underline{\mathbf{L}}_b + \mathbf{E}_b) \mathbf{\Pi}_b,$$

where $\mathbf{E}_b \in \mathbb{R}^{n \times K}$ is obtained by resampling entries of $\mathbf{Z}_{b,I^C}^* - \hat{\mathbf{Z}}_{b,I^C}^*$ and $\mathbf{\Pi}_b \in \mathbb{R}^{n \times P}$ is a random permutation matrix. Alignment and confidence area construction then proceed as in the

nonparametric bootstrap approach, with alignment done using a Procrustes rotation and confidence areas estimated separately for each sample based on the sample mean and covariance across the B bootstrap replicates.

2.3. Compromise Bootstrap

The parametric bootstrap approach requires training only one feature learner $T(\cdot, \hat{\theta})$, so it is more computationally efficient than the nonparametric bootstrap. However, we expect that multiple feature learners may more accurately represent uncertainty. As a compromise, we adapt the parametric bootstrap to the case where S feature extractors are trained, with $1 < S < B$. Specifically, we aim to simulate $\mathbf{L}_1, \dots, \mathbf{L}_B$ in the case where we have access to $T(\cdot, \hat{\theta}_s)$ for $s = 1, \dots, S$. From a high level, this approach applies the parametric bootstrap to each of the S feature learners, resampling residual errors across all.

We begin as in the nonparametric bootstrap, extracting inference set features \mathbf{Z}_{s,I^C} using each of the S trained feature learners. However, instead of directly applying dimensionality reduction and alignment, we simulate additional reduced features as in the parametric approach. Specifically, we compute a rank- K truncated SVD $\hat{\mathbf{Z}}_{s,I^C} = \mathbf{U}_s \boldsymbol{\Sigma}_s \mathbf{V}_s^T$ of \mathbf{Z}_{s,I^C} and define $\underline{\mathbf{L}}_s = \mathbf{U}_s \boldsymbol{\Sigma}_s$. We then generate

$$\mathbf{L}_b = (\underline{\mathbf{L}}_{s(b)} + \mathbf{E}_b) \mathbf{\Pi}_b,$$

where $s(b)$ is drawn uniformly from $1, \dots, S$, \mathbf{E}_b resamples entries across $\mathbf{Z}_{s,I^C} - \hat{\mathbf{Z}}_{s,I^C}$, and $\mathbf{\Pi}_b$ is a random permutation matrix. Given the \mathbf{L}_b , we generate confidence areas for each sample i as before.

3. Simulations

We conduct two simulation studies. The first uses a low-rank model and permits the calculation of coverage rates. However, it is less representative of realistic feature learning settings because the simulated data are matrix-valued and a simple linear combination of latent features. The second generates images using a spatial point process where a small set of parameters control the generative mechanism. The distributed feature learning associated with this setting prevents us from computing the coverage of confidence areas. However, its image data input and nonlinear transformation of features more accurately reflect practice.

3.1. Low-Rank Model

The first simulation generates samples $\mathbf{X} \in \mathbb{R}^{n \times D}$ using,

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T + \mathbf{E} \quad \boldsymbol{\Sigma} = \text{diag}(c\mathbf{1}_K) \quad \mathbf{U} \sim \text{Haar}(n, K) \quad (4)$$

$$y = \mathbf{U}\boldsymbol{\Sigma}\beta + \epsilon \quad \beta = \left(b\mathbf{1}_{\frac{K}{2}}, -b\mathbf{1}_{\frac{K}{2}} \right) \quad \mathbf{V} \sim \text{Haar}(D, K) \quad (5)$$

$$E_{ij} \sim \mathcal{N}(0, \sigma_{\mathbf{E}}^2) \quad (6)$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma_y^2) \quad (7)$$

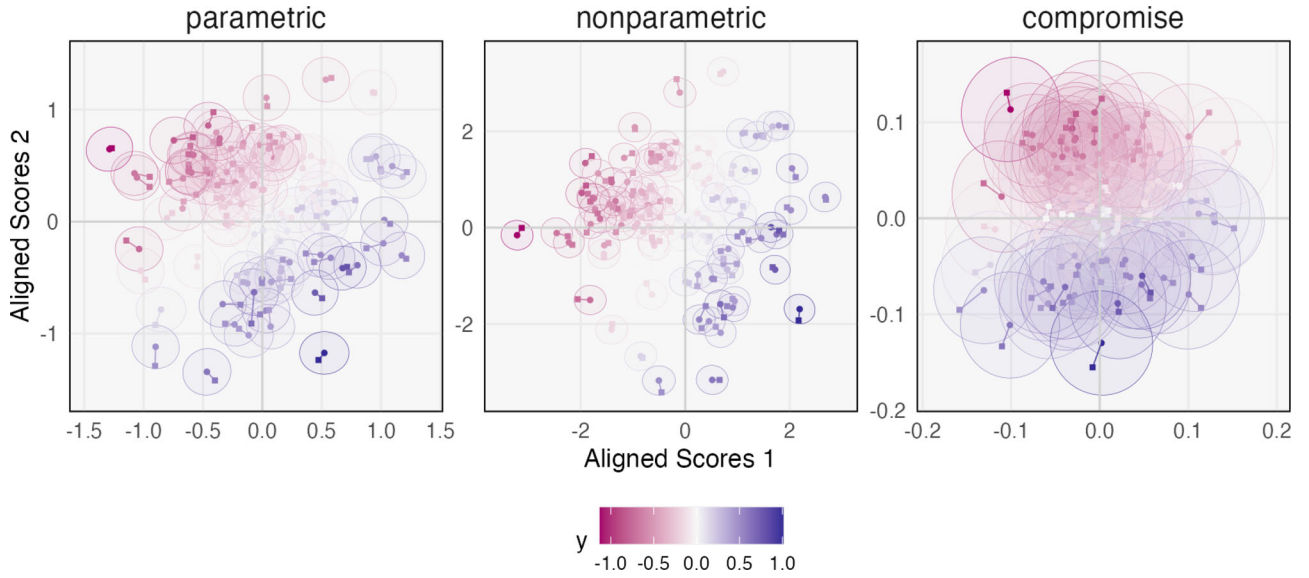


Figure 2. Projections from the low-rank data simulation. Each ellipse gives the confidence area for the latent coordinates of one sample. Squares are the positions of the true low-rank coordinates after Procrustes rotating them to align with the centers of the ellipses. The confidence areas for the nonparametric bootstrap are smaller than those for the parametric bootstrap. Those for the compromise method are conservative.

where $\text{Haar}(n, K)$ denotes a random orthonormal matrix with n rows and K columns. \mathbf{X} is a random rank- K matrix observed with Gaussian noise. y is a response depending on the latent coordinates of each row. The specific parameters we use are $N = 1000$, $D = 100$, $c = 100$, $b = 1$, $K = 2$, $\sigma_E^2 = \sigma_y^2 = 0.1$. Note that this is a model of the data \mathbf{X} , not the features \mathbf{Z} , as in (2).

As a feature extractor, we use a randomly perturbed and permuted estimate of the latent coordinates based on the rank K -truncated SVD,

$$\mathbf{Z} = (\hat{\mathbf{U}}\hat{\Sigma} + \tilde{\mathbf{E}})\mathbf{\Pi} \quad (8)$$

where $\hat{\mathbf{U}}$ and $\hat{\Sigma}$ contain the top K left singular vectors and values of \mathbf{X} and the ij th entry of $\tilde{\mathbf{E}}$ is drawn $\tilde{E}_{ij} \sim \mathcal{N}(0, 0.1^2)$. The permutation $\mathbf{\Pi}$ and noise $\tilde{\mathbf{E}}$ mimic the variation that would be expected across retrained feature extractors. Given the source data \mathbf{X} and extracted features \mathbf{Z} , we apply all three bootstrap methods, generating $B = 1000$ bootstrap replicates in each case. We train $S = 100$ separate feature extractors for the compromise approach.

The resulting 95% confidence ellipses are shown in Figure 2. Qualitatively, the parametric and nonparametric bootstrap approaches provide similar output. A smooth color gradient in values of y suggests that the feature extractors accurately estimate the latent factors. We have Procrustes aligned the underlying coordinates $\mathbf{U}\Sigma$ (squares) with the B bootstrap replicates. The fact that the ellipses contain most squares suggests that the bootstrap accurately reflects uncertainty in the estimated projections. In fact, for the parametric and nonparametric approaches, the empirical coverage rates of these ellipses are 96.4% and 95.2%, respectively. On the other hand, the compromise approach appears to be overly conservative, with a 99.9% coverage. This qualitative relationship between the sizes of confidence area sizes arises in the remaining simulations and data analysis as well, suggesting that though this simulation is a highly simplified setup, it captures some general features of the three proposed bootstrap approaches.

3.2. Spatial Point Process

Our second simulation generates a collection of images using a point process whose parameters vary from image to image. Intuitively, each image represents cells viewed through a microscope, and different latent parameters influence the cell ecosystem. A single response value y is associated with each combination of latent parameters. Figure 3 gives example images for varying y . We generate 10,000 of these $64 \times 64 \times 3$ -dimensional RGB images. In contrast to the previous simulation, which used a simplified SVD-based feature extractor, this dataset requires algorithmic feature learning to transform the image data into vector representations. Therefore, this dataset provides a more realistic setting for comparing the proposed strategies for evaluating embedding variability.

3.2.1. Generation

We sample the locations of new cells using an intensity function drawn from a two-dimensional marked Log Cox Matérn Process (LCMP) (Diggle et al. 2013). Specifically, we apply the following steps. Define the correlation between pixels $x, y \in \mathbb{R}^2$ using a Matérn covariance function,

$$C_{\nu, \alpha}(\|x - y\|) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x - y\|}{\alpha} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|x - y\|}{\alpha} \right), \quad (9)$$

where α acts like a bandwidth parameter and ν controls roughness. Next, simulate a nonnegative intensity function $\Lambda(x)$ along the image grid using $\log \Lambda(x) \sim \mathcal{N}(0, \mathbf{C}_{\nu_\Lambda, \alpha_\Lambda})$, where $\mathbf{C}_{\nu_\Lambda, \alpha_\Lambda}$ is the covariance matrix induced by (9). This function is a baseline intensity that determines the location of cells, regardless of cell type. Our LCMP has R classes (cell types) with relative frequencies governed by additional processes $\log B_r(x) \sim \mathcal{N}(\beta_r, \mathbf{C}_{\nu_B, \alpha_B})$. These processes provide relative frequencies for each class across locations x . The intercept β_r modulates the r th class' frequency across all pixel coordinates. Given these intensity functions, we simulate N cell locations

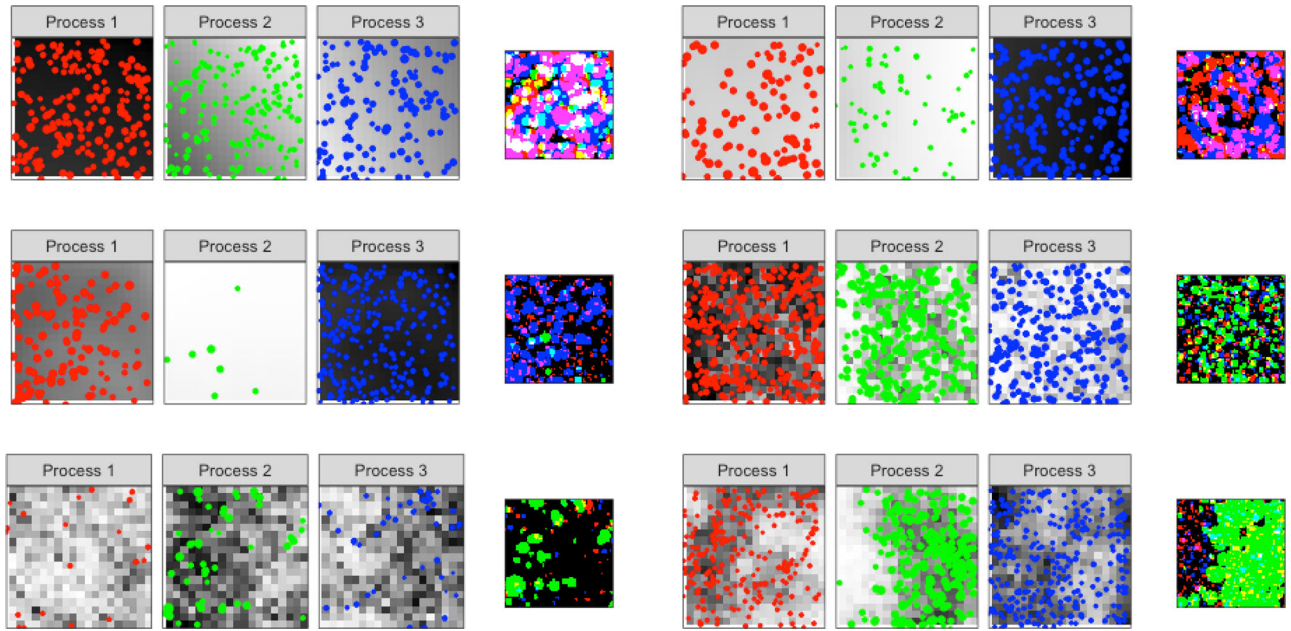


Figure 3. Example images for low (top), average (middle), and high (bottom) values of y_j . Three relative intensity functions $B_r(x)$ are generated for each sample, shown as background heatmaps. Samples drawn from each process are overlaid as circles. The final images combine points across processes, removing the underlying intensity function, which is unavailable to the feature learner. Smaller values of y_j are associated with smoother, less structured intensity functions.

through an inhomogeneous Poisson process with intensity $\Lambda(x)$. For a cell at location x , we assign it to cell type r with probability $\frac{B_r^r(x)}{\sum_{r'=1}^R B_{r'}^r(x)}$. τ is a temperature parameter controlling the degree of mixedness between cell types at a given location.

Finally, we vary the number and sizes of cells. The number of cells per image is drawn uniformly from 50 to 1000. The cells from class R are drawn with a random Gamma $(5, \lambda_r)$ radius. Supplementary Table 1 summarizes all parameters used to generate each image. Each parameter is drawn uniformly within its range, which has been chosen to provide sufficient variation in image appearance. These parameters are the true underlying features associated with the simulated images, giving the most concise description of the observed variation. The response y is a hand-specified linear combination of these parameters and is detailed in Supplementary Section 2.

3.2.2. Feature Learning

We study how the following parameters influence the estimated confidence areas,

- Learning versus inference split sizes: We vary the proportion of data used for learning and inference. We sample I so that $\frac{1}{n}|I| \in \{0.15, 0.5, 0.9\}$.
- Models trained: For feature extractors, we train CNN, VAE, and RCF models on the learning split I .
- Model complexity: We train VAEs whose hidden layer has dimensionality $P \in \{32, 64, 128\}$. Similarly, we vary the number of first-layer convolutional filters in the CNN model across $P \in \{32, 64, 128\}$. For the RCF, we use $P \in \{256, 512, 1024\}$ random features. This increase reflects that more random features must be considered before identifying a subset of predictive ones.

- Bootstrap method: We use the parametric, nonparametric, and compromise bootstrap strategies.

Figure 4 shows the activations of learned features across 2000 images for two perturbed versions of the training data when 90% of the data are used for inference and $P = 64$ (CNN, VAE) and 512 (RCF). Note that, across algorithms, there is no simple correspondence between learned and source features (i.e., parameters of the underlying simulation). Instead, there are clusters of learned features with similar patterns across samples. We also find subsets of features across all models that are only weakly correlated with any source feature. This lack of one-to-one correspondence between true and learned features is consistent with the phenomenon of distributed representations in neural networks (Hinton 1984; Le and Mikolov 2014).

Certain source features appear easier to represent than others because many learned features are strongly correlated with them. For example, many features are correlated with N_i , the total number of cells in the image, and λ_{i1} , which controls the size distribution of the first cell type. Depending on the model, the bandwidth α_{ir} , roughness ν_{ir} , and prevalence β_{ik} parameters are either only weakly or not at all correlated with learned features. Even when features detect variation in α_{ir} and ν_{ir} , they cannot disambiguate between these two parameters. Finally, the CNN and VAE features tend to be more clustered. In contrast, the RCF features show more gradual shifts in correlation strength. They also show only slight variation in correlation strength across features other than λ_{i1} and N_i .

3.2.3. Embedding Variability

Figure 5(a) gives example confidence areas across models and bootstrapping approaches. In contrast to Figure 2 from the low-rank simulation, the areas from the nonparametric bootstrap

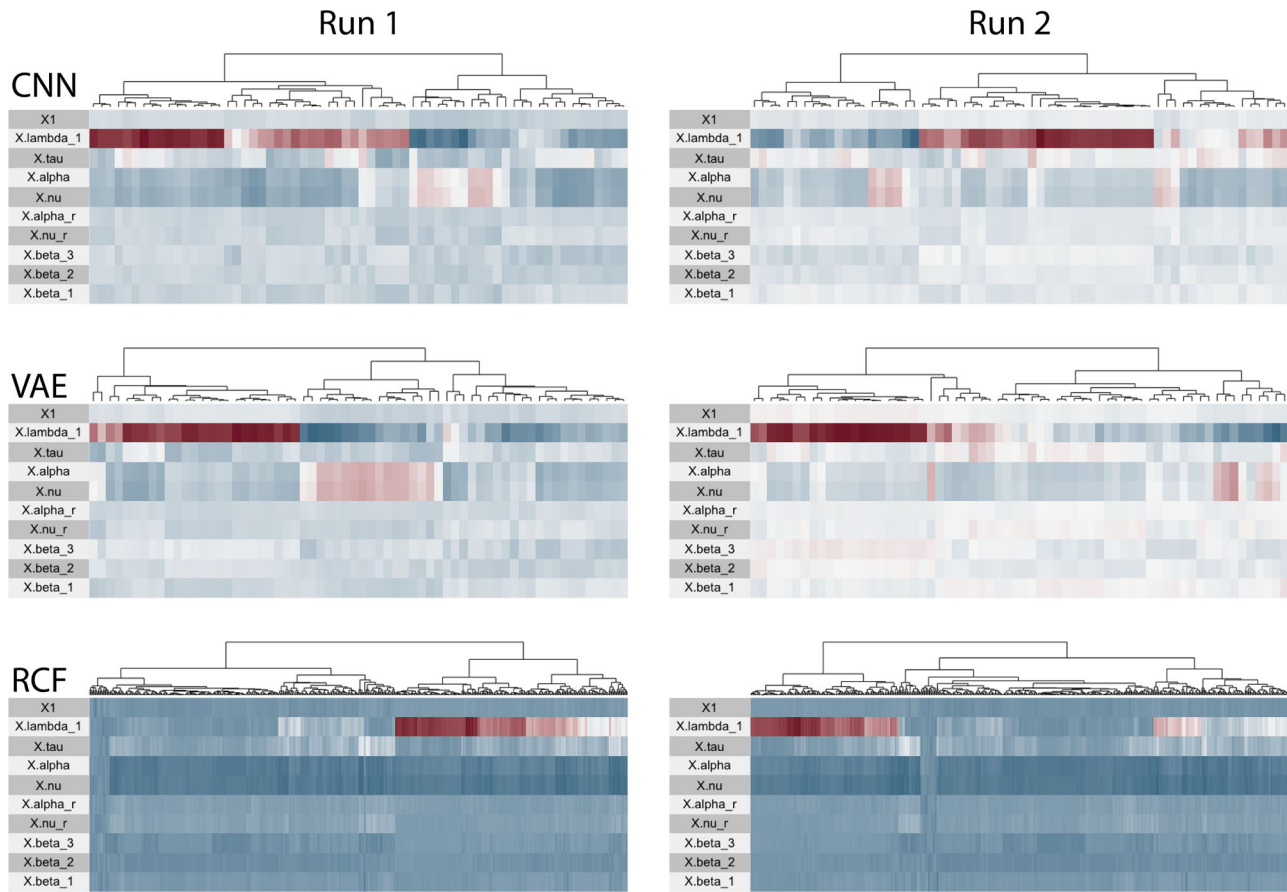


Figure 4. Each feature learning algorithm learns a distributed representation of the true underlying features in the simulation. Within each heatmap, rows correspond to the parameters in Supplementary Table 1. Columns are activations of learned features and have been reordered using the package (Barter and Yu 2018). The shading of a cell gives the correlation between the true and learned features. Darker and lighter colors indicate stronger and weaker correlations, respectively.

are larger than those from the parametric bootstrap. This disagreement suggests that the proposed mechanism of (4) and (8) underestimates variability in learned features in more complex feature learning settings. We recommend using the nonparametric confidence areas whenever a disagreement is observed between nonparametric and parametric bootstrap strategies. These areas are more conservative and reflect variation across runs. As before, the compromise bootstrap has larger confidence areas than either alternative.

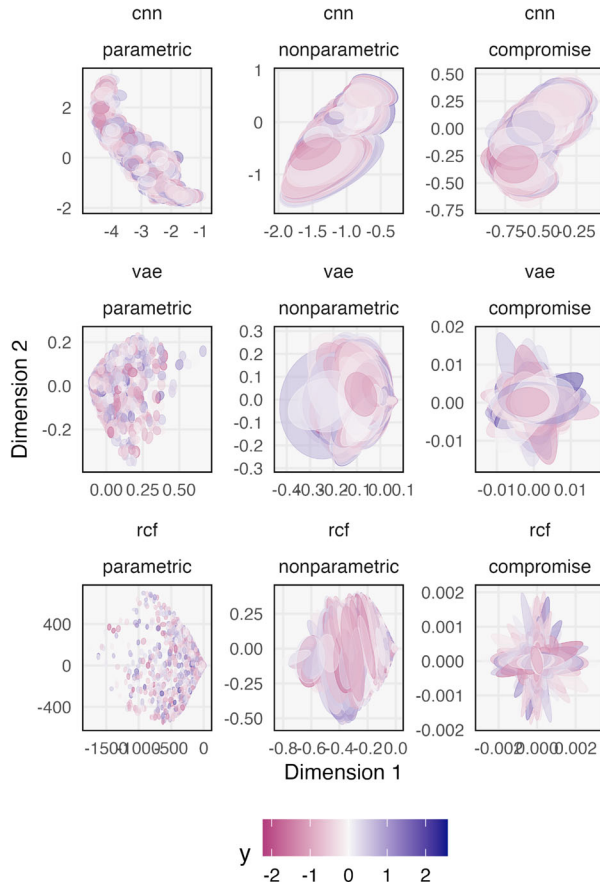
We next analyze how experimental factors influence the qualitative appearance of estimated confidence areas. The RCF tends to have smaller confidence areas than the CNN and VAE. This suggests that shallower models may have more stable learned feature embeddings, an observation with implications for final model selection, depending on the downstream tasks within which these embeddings are used. Figure 5(b) shows confidence areas for a single model (CNN) and bootstrap procedure (parametric) across a range of model complexities and split proportions. For larger P , projections are further from the origin, suggesting larger activations on average. However, the size of each sample's confidence area does not appear to vary across model complexities, and overparameterization may not be a cause for concern. Finally, the fraction of data used for feature learning does not appear to affect the strength of the association with the response or the size of the confidence areas.

Hence, this algorithm hyperparameter does not substantially influence downstream interpretation. Supplementary Section 3 gives corresponding figures for the other models.

4. Data Analysis

To illustrate the application of our proposed methods, we next analyze the spatial proteomics dataset reported in Keren et al. (2018), which found that spatially homogeneous Triple Negative Breast Cancer (TNBC) tumors were associated with more aggressive disease. Classical proteomics methods measure protein expression levels for a collection of cells but cannot specify each cell's location. In contrast, these data provide an image delineating cell boundaries and the underlying protein expression levels. We will work with spatial cell delineations but not protein expression levels. This allows us to study feature learning within the images without linking expression and image data, which is a complex integration problem. Given this simplification, the data are 2048×2048 -dimensional images, one for each of 41 patients. Each pixel has a value of 1 through 7, encoding the cell types associated with each pixel. To ensure that the cell types are treated as categorical, we transform pixels to their one-hot encodings, resulting in a collection of $2048 \times 2048 \times 7$ binary matrices.

A. Varying learning and bootstrap methods



B. Varying splits and model complexity (CNN, parametric)

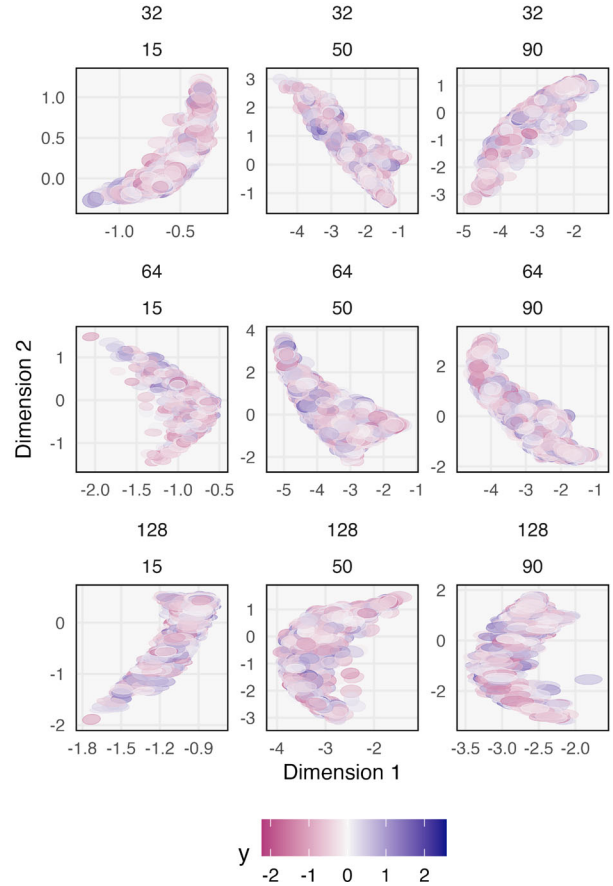


Figure 5. (a) The 95% confidence areas associated with projections from the spatial point process simulation. Each point corresponds to one image. Only the setting with 90% of the data used for feature learning and the midsize models ($P = 64$ for the CNN and VAE, $P = 512$ for the RCF) is shown. (b) A view of confidence areas for the CNN across a range of learning split fractions (0.15, 0.5, 0.9) and model complexities ($P = 32, 64, 128$), all using the nonparametric approach.

4.1. Feature Learning

To set up a prediction problem, we split each image into $512 \times 512 \times 7$ patches. Patches from 32 of the patients are reserved for feature learning. Four among these 32 are used to tune algorithm parameters. As a response variable, we use $y_i = \log\left(\frac{\#\{\text{Tumor cells in } x_i\}}{\#\{\text{Immune cells in } x_i\}}\right)$. These y_i provide the signal for the supervised feature learners. Example cell patches are shown in Figure 6. We fit the same models (CNN, VAE, and RCF) as discussed in Section 3, varying model complexity over the same parameters as before. As a baseline, we compare against a ridge regression with pixel-wise composition features. We train a model with y as the response and the average number of pixels in each cell-type category as a seven-dimensional feature vector. This comparison helps to determine whether the model has learned interesting features for counting cells, like cell size and boundaries, rather than simply averaging across pixel values. Indeed, Figure 7 shows that, except in models with low capacity P , performance is improved when learning features algorithmically.

4.2. Embedding Variability

To characterize features and their uncertainty, we perform $B = 100$ iterations for each of the parametric, nonparametric,

and compromise bootstrap methods applied to samples from the nine patients in the inference set. Figure 8 provides two-dimensional projections for fixed model complexities. All methods learn to differentiate between patches with small and large values of y_i , even the VAE, which is unsupervised. Comparing rows, the RCF appears to give the most stable representations, while the coordinates for the VAE have larger confidence areas in general. Specific axis directions tend to be more uncertain than others, reflected by the eccentricity of ellipses. For example, viewing estimates from the nonparametric approach, the VAE projections have the highest uncertainty for low values of Dimension 2. Analogously, high values of Dimension 2 have high uncertainty in the RCF. In either case, we should be more careful interpreting differences between samples along this direction.

For the CNN and RCF, the three bootstrap approaches give qualitatively similar conclusions about regions with higher and lower embedding precision, though the average sizes of the confidence areas differ. In this application, the size of confidence areas for the compromise approach seems intermediate between the parametric and nonparametric approaches. For the VAE, the bootstrap approaches do not appear to agree. The compromise approach generally gives much larger confidence areas, potentially reflecting a failure of the Procrustes alignment. As in the spatial point process simulation, we find few differences in the

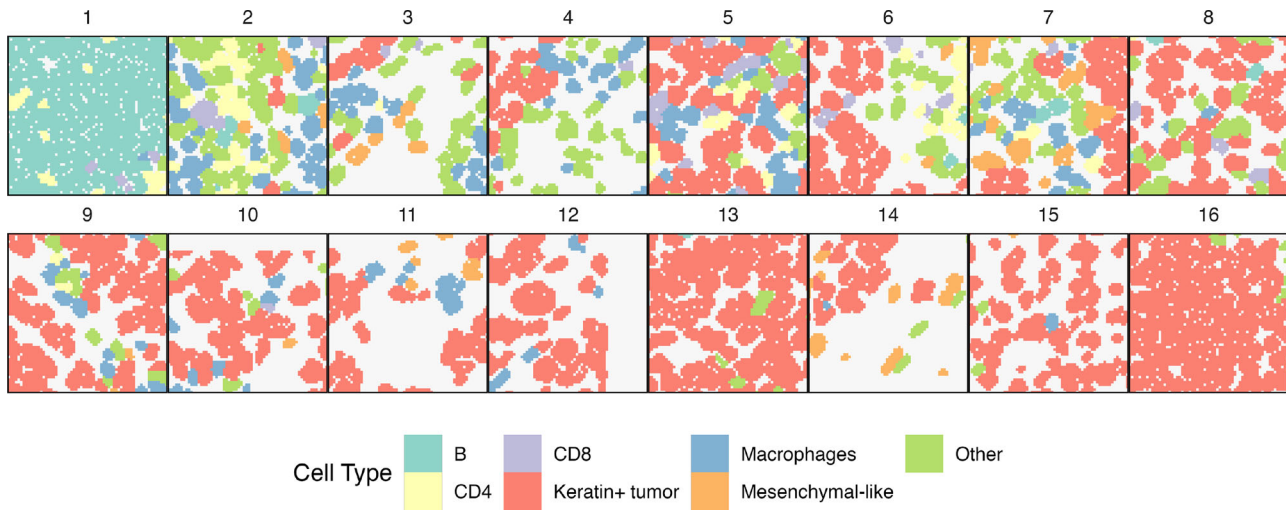


Figure 6. Example patches from the TNBC data. Panels are ordered by y_i , the (log) fraction of cells they contain that belongs to the tumor. This relative fraction of tumor cells provides a signal for the supervised algorithms, whose goal is to place patches from new patients along this gradient correctly.

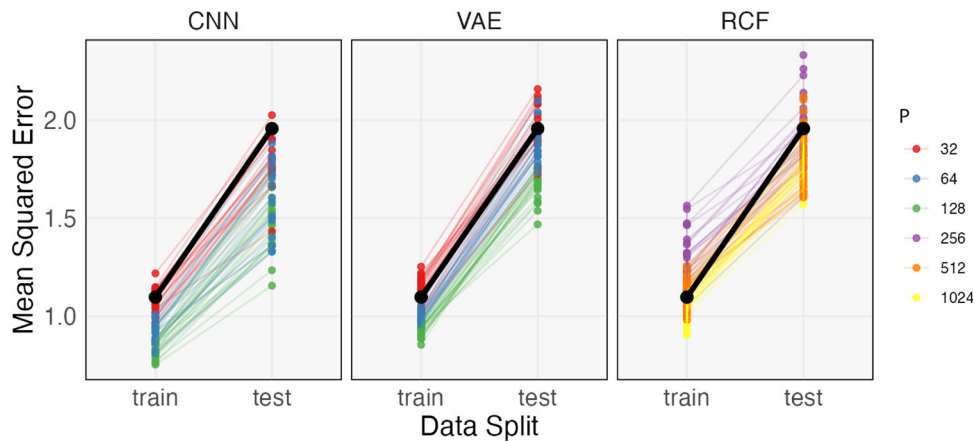


Figure 7. Relative performance of feature learning strategies on the TNBC data. Linked points come from the same bootstrap replicate. The solid black line gives the baseline ridge regression approach using the manually generated raw pixel counts for each cell type as predictors. Predictions from the four patients used to tune each method are omitted; this split has few patches, and the estimated mean squared errors have high variance.

embedding uncertainty across models with different complexities; see Supplementary Figure 10.

Figure 9 overlays example patches onto aligned coordinates. These patches support the interpretation of the learned feature embedding. In the CNN, samples in the bottom right have a high fraction of immune cells, those on the top left are mostly tumor cells, and those at the top right have lower cell density. The confidence areas in Figure 8 suggest that embeddings of tissues with a high fraction of immune cells tend to show more variability across bootstraps. The lower right region of the VAE has high cell diversity, and the upper left has a lower density. Regions with higher cell diversity and larger proportions of immune cells also have less precisely estimated embeddings, which is consistent with the regions of high variability observed in the CNN embeddings. The RCF’s first dimension similarly reflects the tumor versus immune gradient. The smaller confidence areas along this axis suggest that the feature extractor more reliably captures this gradient than any variation across Dimension 2. Overall, deep feature learners appear to have more variable embeddings than the RCF, especially for patches from immune-rich tissue samples. Therefore, if the ultimate goal is a

visual comparison of sources of spatial variation, the RCF may yield more reliable conclusions. If the CNN or VAE are still preferred, their embeddings for immune-rich samples should be treated more skeptically.

5. Discussion

We have adapted approaches to bootstrapping PCA to support the evaluation of embedding variability in the algorithmic feature learning context. To contrast these proposals, we have used two simulation studies. Their experimental results suggest that, in more complex settings, a parametric bootstrap based on a single set of learned features does not reflect the degree of uncertainty present when comparing features derived from independently trained models. In contrast, a nonparametric approach yielded more realistic confidence areas. In addition to comparing alternative bootstrap methods, we evaluated the influence of data and model characteristics on the estimated confidence areas. We found that algorithms could differ dramatically in the size of their confidence areas but that model complexity within a single algorithm has little impact.

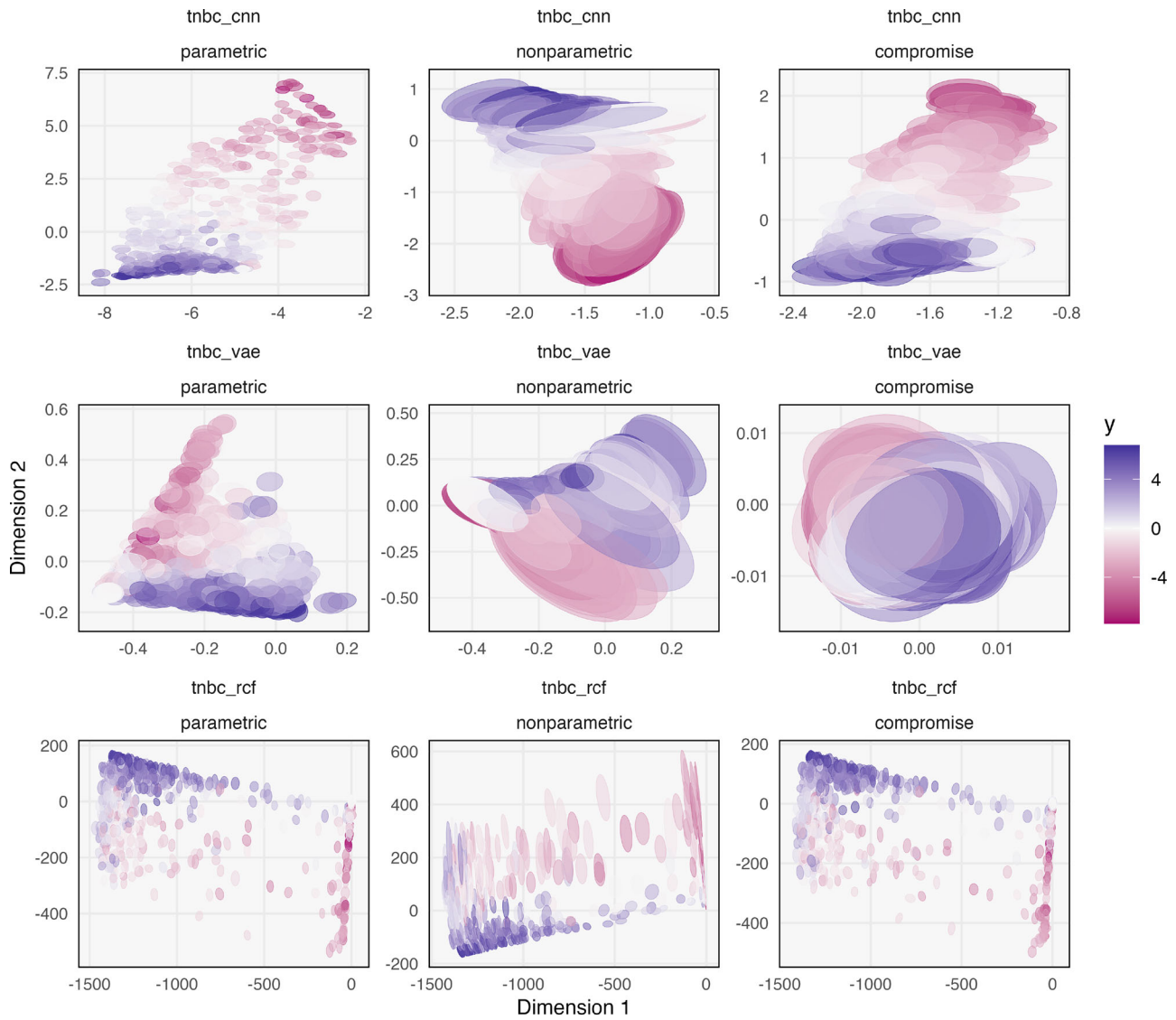


Figure 8. Confidence areas from the TNBC application. Points are shaded by $y_i = \log\left(\frac{\#\{\text{Tumor cells in } x_i\}}{\#\{\text{Immune cells in } x_i\}}\right)$, which provides the supervisory signal to the CNN and RCF during feature extraction. Models and bootstrap procedures are arranged along rows and columns, respectively. Only the models with intermediate complexity ($P = 64$ for the CNN and VAE, $P = 512$ for the RCF) are shown. Analogous figures for other P are given in the supplementary materials.

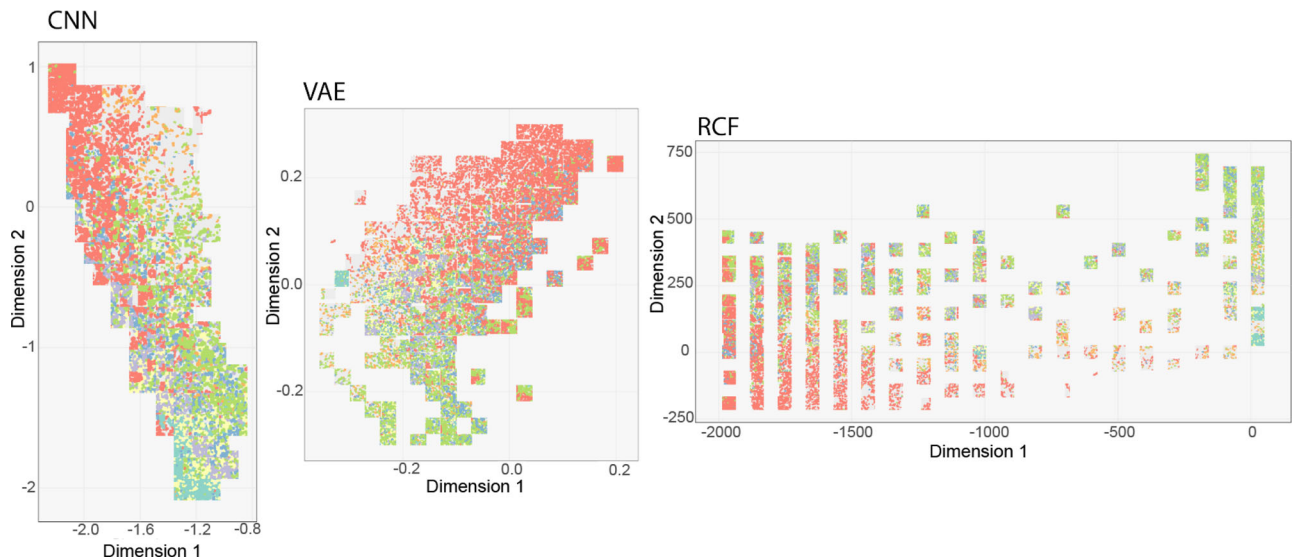


Figure 9. A version of the nonparametric bootstrap column from the embeddings in Figure 8, overlaying representative samples across the learned feature space. Cells are drawn as in Figure 6. The overall shape of the region in which images are displayed mirrors the shape of the cloud of points in Figure 8.

Our empirical results raise several questions for further study. First, though we have found coverage rates for the parametric and nonparametric bootstraps to be acceptable in the low-rank simulation, we have yet to study the theoretical properties of these procedures. A theoretical guarantee would be worthwhile, and it could clarify differences that arise in projection uncertainties between bootstrap and feature learning approaches. A related challenge is exploring the alignment method's influence on the resulting bootstrap regions. We have relied on a simple Procrustes rotation to align the principal subspaces learned by different bootstrap replicates of complex feature learning algorithms. More powerful alternatives could be considered for both dimensionality reduction and alignment. Further, for the CNN and RCF models in the data analysis example, we find that the sizes of confidence areas for the compromise approach are intermediate between those for the parametric and nonparametric bootstraps. However, in both simulations, these confidence areas tend to be larger, and the method is unnecessarily conservative. A better understanding of how the resampled residuals E_{ij} are influenced by the presence of multiple feature learners could yield improved implementations of the compromise bootstrap method.

As algorithmic feature learning methods play more prominent roles in scientific problems with non-matrix structured data, it is natural to attempt to quantify the precision with which the associated learned representations are estimated, as discussed here. More generally, uncertainty can propagate to downstream tasks that depend on these learned features. This uncertainty may depend on the data modality and learning algorithms used. Calibrating inferences to account for representational uncertainty across data modalities and analytical workflows presents an important area for future work.

Supplementary Materials

Appendix: A PDF with additional supporting materials. Includes sections describing details of the spatial point process simulation setup and explaining how to access data and reproduce all analysis. Also provides supplementary figures referred to within the main manuscript. (supplement.pdf, PDF document)

Acknowledgments

The author thanks Susan Holmes, Karl Rohe, three reviewers, the associate editor, and the editor for feedback which improved the manuscript. Research was performed with assistance of the UW-Madison Center For High Throughput Computing (CHTC).

References

Barter, R. L., and Yu, B. (2018), "Superheat: An R Package for Creating Beautiful and Extendable Heatmaps for Visualizing Complex Data," *Journal of Computational and Graphical Statistics*, 27, 910–922. [7]

- Chateau, F., and Lebart, L. (1996), "Assessing Sample Variability in the Visualization Techniques Related to Principal Component Analysis: Bootstrap and Alternative Simulation Methods," in *Compstat*, eds. D. Edwards, N. E. Raun, pp. 205–210, Heidelberg: Springer. [1,2]
- Diggle, P. J., Moraga, P., Rowlingson, B., and Taylor, B. M. (2013), "Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm," *Statistical Science*, 28, 542–563. [5]
- Elguero, E., and Holmes-Junca, S. (1988), "Confidence Regions for Projection Pursuit Density Estimates," in *Compstat*, eds. D. Edwards, N. E. Raun, pp. 59–63, Heidelberg: Springer. [1,2]
- Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009), "Visualizing Higher-Layer Features of a Deep Network," University of Montreal, 1341, 1. [1]
- Friedman, J., Hastie, T., and Tibshirani, R. (2001), *The Elements of Statistical Learning* (Vol. 1) (No. 10). Springer Series in Statistics, New York: Springer. [2]
- Gross, S. M., Taylor, J., and Tibshirani, R. (2015), "A Selective Approach to Internal Inference," arXiv preprint arXiv:1510.00486. [1,3]
- Hinton, G. E. (1984), "Distributed Representations," Technical Report CMU-CS-84-157. [6]
- Josse, J., Wager, S., and Husson, F. (2016), "Confidence Areas for Fixed-Effects PCA," *Journal of Computational and Graphical Statistics*, 25, 28–48. [1,2,4]
- Ke, Z. T., and Wang, M. (2022), "Using SVD for Topic Modeling," *Journal of the American Statistical Association*, 1–16. DOI:10.1080/01621459.2022.2123813 [1]
- Keren, L., Bosse, M., Marquez, D., Angoshtari, R., Jain, S., Varma, S., et al. (2018), "A Structured Tumor-Immune Microenvironment in Triple Negative Breast Cancer Revealed by Multiplexed Ion Beam Imaging," *Cell*, 174, 1373–1387. [7]
- Kim, K., Li, B., Yu, Z., and Li, L. (2020), "On Post Dimension Reduction Statistical Inference," *The Annals of Statistics*, 48, 1567–1592. [1]
- Kingma, D. P., and Welling, M. (2014), "Auto-Encoding Variational Bayes," *2nd International Conference on Learning Representations, ICLR 2014 Conference Track Proceedings*. [2]
- Le, Q., and Mikolov, T. (2014), "Distributed Representations of Sentences and Documents," in *International Conference on Machine Learning*, pp. 1188–1196. [6]
- Nguyen, A., Yosinski, J., and Clune, J. (2019), "Understanding Neural Networks via Feature Visualization: A Survey," in *Explainable ai: Interpreting, Explaining and Visualizing Deep Learning*, eds. W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, K.-R. Müller, pp. 55–76, Cham: Springer. [1]
- Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017), "Svcca: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability," in *Advances in Neural Information Processing Systems*, pp. 6076–6085. [2]
- Rahimi, A., and Recht, B. (2008), "Weighted Sums of Random Kitchen Sinks: Replacing Minimization with Randomization in Learning," *Advances in Neural Information Processing Systems*, 21, 1313–1320. [2]
- Van Den Oord, A., and Vinyals, O. (2017), "Neural Discrete Representation Learning," in *Advances in Neural Information Processing Systems*, pp. 6306–6315. [2]
- Wang, S., McCormick, T. H., and Leek, J. T. (2020), "Methods for Correcting Inference based on Outcomes Predicted by Machine Learning," *Proceedings of the National Academy of Sciences*, 117, 30266–30275. [1,3]