MAT 280: Harmonic Analysis on Graphs & Networks
Lecture 12: Applications of Sparse Graphs: Signal Classification Using Sparse Representation

Naoki Saito

Department of Mathematics
University of California, Davis

November 5, 2019

Outline

Outline

Classification Basics

Let $\mathscr{X} := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of data samples in $\mathbb{R}^m$ with class structure, i.e., each $\boldsymbol{x}_i$ has a *class label* from the set $\{1, \ldots, L\}$, $1 < L \leq N$.

Setup:

- Set aside a *training set* $\mathscr{X}_{\text{tr}} \subset \mathscr{X}$ where the class label of each sample is known and available
- Goal: Classify (predict the labels of) the remaining samples in the *test set* $\mathscr{X}_{\text{te}} := \mathscr{X} \setminus \mathscr{X}_{\text{tr}}$

Examples:

- Face recognition: Classes are subjects (people)
- Automated reading of postal zip codes: Classes are the digits $0, 1, \ldots, 9$.

Classification Basics

Let $\mathscr{X} := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of data samples in $\mathbb{R}^m$ with class structure, i.e., each $\boldsymbol{x}_i$ has a *class label* from the set $\{1, \ldots, L\}$, $1 < L \leq N$.

Setup:

- Set aside a *training set* $\mathscr{X}_{\text{tr}} \subset \mathscr{X}$ where the class label of each sample is known and available
- Goal: Classify (predict the labels of) the remaining samples in the *test set* $\mathscr{X}_{\text{te}} := \mathscr{X} \setminus \mathscr{X}_{\text{tr}}$

Examples:

- Face recognition: Classes are subjects (people)
- Automated reading of postal zip codes: Classes are the digits $0, 1, \ldots, 9$.

Classification Basics

Let $\mathscr{X} := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of data samples in $\mathbb{R}^m$ with class structure, i.e., each $\boldsymbol{x}_i$ has a *class label* from the set $\{1, \ldots, L\}$, $1 < L \leq N$.

Setup:

- Set aside a *training set* $\mathscr{X}_{\rm tr} \subset \mathscr{X}$ where the class label of each sample is known and available
- Goal: Classify (predict the labels of) the remaining samples in the *test set* $\mathscr{X}_{\rm te} := \mathscr{X} \setminus \mathscr{X}_{\rm tr}$

Examples:

- Face recognition: Classes are subjects (people)
- Automated reading of postal zip codes: Classes are the digits $0, 1, \ldots, 9$.

Some Common Assumptions

- The *intrinsic dimension* of the data d is much lower than the ambient dimension m: $d \ll m$

  Example: The space of all face images

- The data in each class lie on a so-called *class manifold*:
  - Smooth
  - Might intersect other class manifolds
  - Could be highly nonlinear

Some Common Assumptions

- The *intrinsic dimension* of the data d is much lower than the ambient dimension m: $d \ll m$

  Example: The space of all face images

- The data in each class lie on a so-called *class manifold*:
  - Smooth
  - Might intersect other class manifolds
  - Could be highly nonlinear

Some Common Assumptions

- The *intrinsic dimension* of the data d is much lower than the ambient dimension m: $d \ll m$

  Example: The space of all face images

- The data in each class lie on a so-called *class manifold*:
  - Smooth
  - Might intersect other class manifolds
  - Could be highly nonlinear

Some Common Assumptions

- The *intrinsic dimension* of the data d is much lower than the ambient dimension m: $d \ll m$

  Example: The space of all face images

- The data in each class lie on a so-called *class manifold*:
  - Smooth
  - Might intersect other class manifolds
  - Could be highly nonlinear

Some Common Assumptions

- The *intrinsic dimension* of the data d is much lower than the ambient dimension m: $d \ll m$

  Example: The space of all face images

- The data in each class lie on a so-called *class manifold*:
  - Smooth
  - Might intersect other class manifolds
  - Could be highly nonlinear

Outline

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\mathrm{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\mathrm{tr}}}] \in \mathbb{R}^{m \times N_{\mathrm{tr}}}.$$

  - Each $\boldsymbol{x}_j$ is normalized to have $\|\boldsymbol{x}_j\|_2 = 1$.
  - $m < N_{\mathrm{tr}}$
- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\mathrm{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\mathrm{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.
- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\mathrm{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\mathrm{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\text{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{tr}}}] \in \mathbb{R}^{m \times N_{\text{tr}}}.$$

  - Each $\boldsymbol{x}_i$ is normalized to have $\|\boldsymbol{x}_i\|_2 = 1$.
  - $m < N_{\text{tr}}$

- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.

- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\text{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{tr}}}] \in \mathbb{R}^{m \times N_{\text{tr}}}.$$

  - Each $\boldsymbol{x}_i$ is normalized to have $\|\boldsymbol{x}_i\|_2 = 1$.
  - $m < N_{\text{tr}}$

- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.

- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\text{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{tr}}}] \in \mathbb{R}^{m \times N_{\text{tr}}}.$$

  - Each $\boldsymbol{x}_i$ is normalized to have $\|\boldsymbol{x}_i\|_2 = 1$.
  - $m < N_{\text{tr}}$

- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.

- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\mathrm{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\mathrm{tr}}}] \in \mathbb{R}^{m \times N_{\mathrm{tr}}}.$$

  - Each $\boldsymbol{x}_i$ is normalized to have $\|\boldsymbol{x}_i\|_2 = 1$.
  - $m < N_{\mathrm{tr}}$

- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\mathrm{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\mathrm{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.

- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\mathrm{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\mathrm{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- Proposed by Wright, Yang, Ganesh, Sastry, and Ma in 2009
- Begin with the *dictionary* of training samples:

$$X_{\text{tr}} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_{\text{tr}}}] \in \mathbb{R}^{m \times N_{\text{tr}}}.$$

  - Each $\boldsymbol{x}_i$ is normalized to have $\|\boldsymbol{x}_i\|_2 = 1$.
  - $m < N_{\text{tr}}$

- Goal: Write the test sample $\boldsymbol{y} \in \mathbb{R}^m$ as a linear combination of the training samples using as few samples as possible:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_0 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(NP-Hard)}$$

  where $\|\boldsymbol{\beta}\|_0$ is a *quasinorm* counting the number of nonzero entries of $\boldsymbol{\beta}$.

- In reality: Finding the sparsest linear combination is not tractable! Approximate it by finding the linear combination whose coefficient vector has *the smallest ℓ^1-norm*:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^{N_{\text{tr}}}} \|\boldsymbol{\beta}\|_1 \text{ subject to } X_{\text{tr}}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{(LP)}$$

Sparse Representation-Based Classification (SRC)

- As real-world data is often corrupted by noise, it is often more practical to solve

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{N_{\text{tr}}}} \left\{ \|\boldsymbol{y} - X_{\text{tr}}\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1 \right\},$$

  where $\lambda > 0$ is the trade off between accuracy in the approximation and sparsity of the coefficient vector.

- This form of the problem formulation is called the *Lasso* estimation; see the excellent book by Hastie, Tibshirani, and Wainwright for the details!

- The test sample $\boldsymbol{y}$ is assigned to the class whose training samples contribute the most to the sparsest approximation of $\boldsymbol{y}$ over the entire training set:

$$\text{class\_label}(\boldsymbol{y}) = \arg\min_{l} \|\boldsymbol{y} - X_{\text{tr}}\delta_l(\boldsymbol{\beta}^*)\|_2,$$

  where $\delta_l(\boldsymbol{\beta}^*)$ is the result of setting all components of $\boldsymbol{\beta}^*$ except those corresponding to class l to zero.

Sparse Representation-Based Classification (SRC)

- As real-world data is often corrupted by noise, it is often more practical to solve

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{N_{\text{tr}}}} \left\{ \|\boldsymbol{y} - X_{\text{tr}}\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1 \right\},$$

  where $\lambda > 0$ is the trade off between accuracy in the approximation and sparsity of the coefficient vector.

- This form of the problem formulation is called the *Lasso* estimation; see the excellent book by Hastie, Tibshirani, and Wainwright for the details!

- The test sample $\boldsymbol{y}$ is assigned to the class whose training samples contribute the most to the sparsest approximation of $\boldsymbol{y}$ over the entire training set:

$$\text{class\_label}(\boldsymbol{y}) = \arg\min_{l} \|\boldsymbol{y} - X_{\text{tr}}\delta_l(\boldsymbol{\beta}^*)\|_2,$$

  where $\delta_l(\boldsymbol{\beta}^*)$ is the result of setting all components of $\boldsymbol{\beta}^*$ except those corresponding to class l to zero.

Sparse Representation-Based Classification (SRC)

- As real-world data is often corrupted by noise, it is often more practical to solve

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{N_{\text{tr}}}} \left\{ \|\boldsymbol{y} - X_{\text{tr}}\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1 \right\},$$

  where $\lambda > 0$ is the trade off between accuracy in the approximation and sparsity of the coefficient vector.

- This form of the problem formulation is called the *Lasso* estimation; see the excellent book by Hastie, Tibshirani, and Wainwright for the details!

- The test sample $\boldsymbol{y}$ is assigned to the class whose training samples contribute the most to the sparsest approximation of $\boldsymbol{y}$ over the entire training set:

$$\text{class\_label}(\boldsymbol{y}) = \arg\min_{l} \|\boldsymbol{y} - X_{\text{tr}}\delta_l(\boldsymbol{\beta}^*)\|_2,$$

  where $\delta_l(\boldsymbol{\beta}^*)$ is the result of setting all components of $\boldsymbol{\beta}^*$ except those corresponding to class l to zero.

Weaknesses in SRC

- SRC assumes that class manifolds are approximately *linear* subspaces that are spanned by their training samples. Yet, class manifolds for facial images under varying poses and expressions are generally *nonlinear*.
- If we can sparsely approximate $\boldsymbol{y}$ using more than one class, then SRC may not be able to distinguish these classes, which can happen when $\boldsymbol{y}$ is close to more than one class manifold.
- The high computational complexity of the ℓ^1-minimization algorithm means that *feature extraction/dimension reduction* is necessary. Yet some discriminant information may be lost in this process, especially if the class manifolds are sparsely sampled.

Weaknesses in SRC

- SRC assumes that class manifolds are approximately *linear* subspaces that are spanned by their training samples. Yet, class manifolds for facial images under varying poses and expressions are generally *nonlinear*.
- If we can sparsely approximate $\boldsymbol{y}$ using more than one class, then SRC may not be able to distinguish these classes, which can happen when $\boldsymbol{y}$ is close to more than one class manifold.
- The high computational complexity of the ℓ^1-minimization algorithm means that *feature extraction/dimension reduction* is necessary. Yet some discriminant information may be lost in this process, especially if the class manifolds are sparsely sampled.

Weaknesses in SRC

- SRC assumes that class manifolds are approximately *linear* subspaces that are spanned by their training samples. Yet, class manifolds for facial images under varying poses and expressions are generally *nonlinear*.
- If we can sparsely approximate $\boldsymbol{y}$ using more than one class, then SRC may not be able to distinguish these classes, which can happen when $\boldsymbol{y}$ is close to more than one class manifold.
- The high computational complexity of the ℓ^1-minimization algorithm means that *feature extraction/dimension reduction* is necessary. Yet some discriminant information may be lost in this process, especially if the class manifolds are sparsely sampled.

Outline

Proposed Algorithm: Overview

Our proposed classification algorithm, *Local Principal Component Analysis SRC* (LPCA-SRC), aims to address these issues by

- Generating *new* training samples in order to *increase the sampling density*;
- Adding a degree of *locality* to SRC, so that only nearby training samples can contribute to the classification of the given test sample.

LPCA-SRC has two parts:

Offline phase: A new set of training samples is generated by finding a basis for the approximate tangent hyperplane at each training sample using *local PCA*. These new training samples are shifted and scaled versions of these tangent plane basis vectors.

Online phase: The matrix/dictionary of (new and original) training samples is *pruned to include only the original training samples that are close to the test sample (with respect to Euclidean distance), and their tangent plane basis vectors.*

Proposed Algorithm: Overview

Our proposed classification algorithm, *Local Principal Component Analysis SRC* (LPCA-SRC), aims to address these issues by

- Generating *new* training samples in order to *increase the sampling density*;
- Adding a degree of *locality* to SRC, so that only nearby training samples can contribute to the classification of the given test sample.

LPCA-SRC has two parts:

Offline phase: A new set of training samples is generated by finding a basis for the approximate tangent hyperplane at each training sample using *local PCA*. These new training samples are shifted and scaled versions of these tangent plane basis vectors.

Online phase: The matrix/dictionary of (new and original) training samples is *pruned to include only the original training samples that are close to the test sample (with respect to Euclidean distance), and their tangent plane basis vectors.*

Proposed Algorithm: Overview

Our proposed classification algorithm, *Local Principal Component Analysis SRC* (LPCA-SRC), aims to address these issues by

- Generating *new* training samples in order to *increase the sampling density*;
- Adding a degree of *locality* to SRC, so that only nearby training samples can contribute to the classification of the given test sample.

LPCA-SRC has two parts:

Offline phase: A new set of training samples is generated by finding a basis for the approximate tangent hyperplane at each training sample using *local PCA*. These new training samples are shifted and scaled versions of these tangent plane basis vectors.

Online phase: The matrix/dictionary of (new and original) training samples is *pruned* to include only the original training samples that are close to the test sample (with respect to Euclidean distance), and their tangent plane basis vectors.

Proposed Algorithm: Overview

Our proposed classification algorithm, *Local Principal Component Analysis SRC* (LPCA-SRC), aims to address these issues by

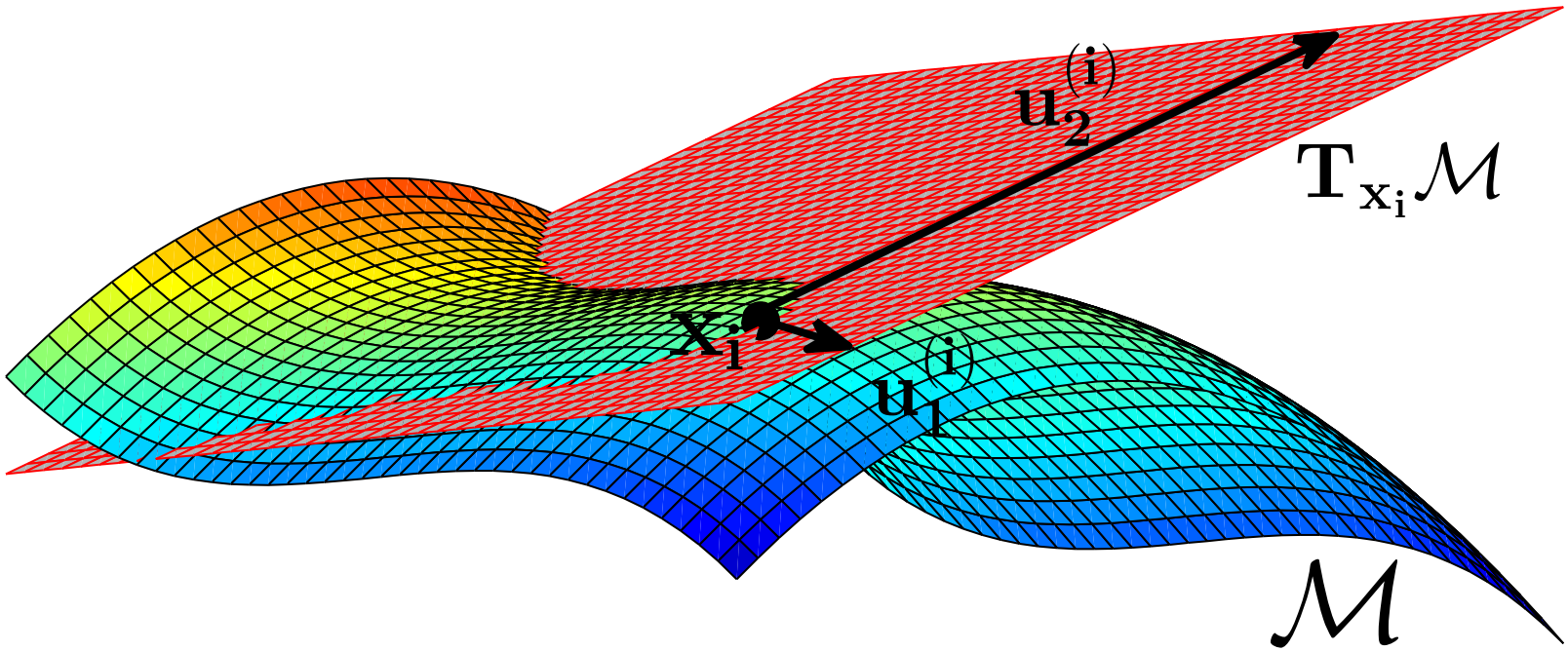
- Generating *new* training samples in order to *increase the sampling density*;
- Adding a degree of *locality* to SRC, so that only nearby training samples can contribute to the classification of the given test sample.

LPCA-SRC has two parts:

Offline phase: A new set of training samples is generated by finding a basis for the approximate tangent hyperplane at each training sample using *local PCA*. These new training samples are shifted and scaled versions of these tangent plane basis vectors.

Online phase: The matrix/dictionary of (new and original) training samples is *pruned to include only the original training samples that are close to the test sample (with respect to Euclidean distance), and their tangent plane basis vectors.*

Illustration of Tangent Vectors



LPCA-SRC Algorithm

Offline Phase: For each class l, and for each normalized training sample $\boldsymbol{x}_i^{(l)}$ in class l,

1. Find the tangent plane basis vectors $\boldsymbol{u}_1^{(l,i)}, \ldots, \boldsymbol{u}_d^{(l,i)}$ at $\boldsymbol{x}_i^{(l)}$ using local PCA.
2. Save the matrix $\tilde{X}^{(l,i)} := [\boldsymbol{x}_i^{(l)}, c\boldsymbol{u}_1^{(l,i)} + \boldsymbol{x}_i^{(l)}, \ldots, c\boldsymbol{u}_d^{(l,i)} + \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{m \times (d+1)}$, where $c = r\gamma$, $\gamma \sim \mathrm{unif}(0,1)$, r is the *estimated neighborhood radius*.

Online Phase: For a given normalized test sample $\boldsymbol{y}$,

1. Initialize the dictionary $D_{\boldsymbol{y}} = \emptyset$.
2. For each $\boldsymbol{x}_i^{(l)}$, check to see if $\|\boldsymbol{y} - \boldsymbol{x}_i^{(l)}\|_2 \le r$ or $\|\boldsymbol{y} + \boldsymbol{x}_i^{(l)}\|_2 \le r$. If either is true, add $\tilde{X}^{(l,i)}$ to the dictionary: $D_{\boldsymbol{y}} \leftarrow [D_{\boldsymbol{y}}, \tilde{X}^{(l,i)}]$.
3. Perform SRC, replacing the dictionary X_{tr} with $D_{\boldsymbol{y}}$. Now the dictionary contains only nearby training samples and the (shifted and scaled) basis vectors of their local tangent planes.

LPCA-SRC Algorithm

Offline Phase: For each class l, and for each normalized training sample $\boldsymbol{x}_i^{(l)}$ in class l,

1. Find the tangent plane basis vectors $\boldsymbol{u}_1^{(l,i)}, \ldots, \boldsymbol{u}_d^{(l,i)}$ at $\boldsymbol{x}_i^{(l)}$ using local PCA.
2. Save the matrix $\tilde{X}^{(l,i)} := [\boldsymbol{x}_i^{(l)}, c\boldsymbol{u}_1^{(l,i)} + \boldsymbol{x}_i^{(l)}, \ldots, c\boldsymbol{u}_d^{(l,i)} + \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{m \times (d+1)}$, where $c = r\gamma$, $\gamma \sim \mathrm{unif}(0,1)$, r is the *estimated neighborhood radius*.

Online Phase: For a given normalized test sample $\boldsymbol{y}$,

1. Initialize the dictionary $D_{\boldsymbol{y}} = \emptyset$.
2. For each $\boldsymbol{x}_i^{(l)}$, check to see if $\|\boldsymbol{y} - \boldsymbol{x}_i^{(l)}\|_2 \le r$ or $\|\boldsymbol{y} + \boldsymbol{x}_i^{(l)}\|_2 \le r$. If either is true, add $\tilde{X}^{(l,i)}$ to the dictionary: $D_{\boldsymbol{y}} \leftarrow [D_{\boldsymbol{y}}, \tilde{X}^{(l,i)}]$.
3. Perform SRC, replacing the dictionary X_{tr} with $D_{\boldsymbol{y}}$. Now the dictionary contains only nearby training samples and the (shifted and scaled) basis vectors of their local tangent planes.

LPCA-SRC Algorithm

Offline Phase: For each class l, and for each normalized training sample $\boldsymbol{x}_i^{(l)}$ in class l,

1. Find the tangent plane basis vectors $\boldsymbol{u}_1^{(l,i)}, \ldots, \boldsymbol{u}_d^{(l,i)}$ at $\boldsymbol{x}_i^{(l)}$ using local PCA.
2. Save the matrix $\tilde{X}^{(l,i)} := [\boldsymbol{x}_i^{(l)}, c\boldsymbol{u}_1^{(l,i)} + \boldsymbol{x}_i^{(l)}, \ldots, c\boldsymbol{u}_d^{(l,i)} + \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{m \times (d+1)}$, where $c = r\gamma$, $\gamma \sim \text{unif}(0,1)$, r is the *estimated neighborhood radius*.

Online Phase: For a given normalized test sample $\boldsymbol{y}$,

1. Initialize the dictionary $D_{\boldsymbol{y}} = \emptyset$.
2. For each $\boldsymbol{x}_i^{(l)}$, check to see if $\|\boldsymbol{y} - \boldsymbol{x}_i^{(l)}\|_2 \le r$ or $\|\boldsymbol{y} + \boldsymbol{x}_i^{(l)}\|_2 \le r$. If either is true, add $\tilde{X}^{(l,i)}$ to the dictionary: $D_{\boldsymbol{y}} \leftarrow [D_{\boldsymbol{y}}, \tilde{X}^{(l,i)}]$.
3. Perform SRC, replacing the dictionary X_{tr} with $D_{\boldsymbol{y}}$. Now the dictionary contains only nearby training samples and the (shifted and scaled) basis vectors of their local tangent planes.

LPCA-SRC Algorithm

Offline Phase: For each class l, and for each normalized training sample $\boldsymbol{x}_i^{(l)}$ in class l,

1. Find the tangent plane basis vectors $\boldsymbol{u}_1^{(l,i)}, \ldots, \boldsymbol{u}_d^{(l,i)}$ at $\boldsymbol{x}_i^{(l)}$ using local PCA.
2. Save the matrix $\tilde{X}^{(l,i)} := [\boldsymbol{x}_i^{(l)}, c\boldsymbol{u}_1^{(l,i)} + \boldsymbol{x}_i^{(l)}, \ldots, c\boldsymbol{u}_d^{(l,i)} + \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{m \times (d+1)}$, where $c = r\gamma$, $\gamma \sim \text{unif}(0,1)$, r is the *estimated neighborhood radius*.

Online Phase: For a given normalized test sample $\boldsymbol{y}$,

1. Initialize the dictionary $D_{\boldsymbol{y}} = \emptyset$.
2. For each $\boldsymbol{x}_i^{(l)}$, check to see if $\|\boldsymbol{y} - \boldsymbol{x}_i^{(l)}\|_2 \le r$ or $\|\boldsymbol{y} + \boldsymbol{x}_i^{(l)}\|_2 \le r$. If either is true, add $\tilde{X}^{(l,i)}$ to the dictionary: $D_{\boldsymbol{y}} \leftarrow [D_{\boldsymbol{y}}, \tilde{X}^{(l,i)}]$.
3. Perform SRC, replacing the dictionary X_{tr} with $D_{\boldsymbol{y}}$. Now the dictionary contains only nearby training samples and the (shifted and scaled) basis vectors of their local tangent planes.

LPCA-SRC Algorithm

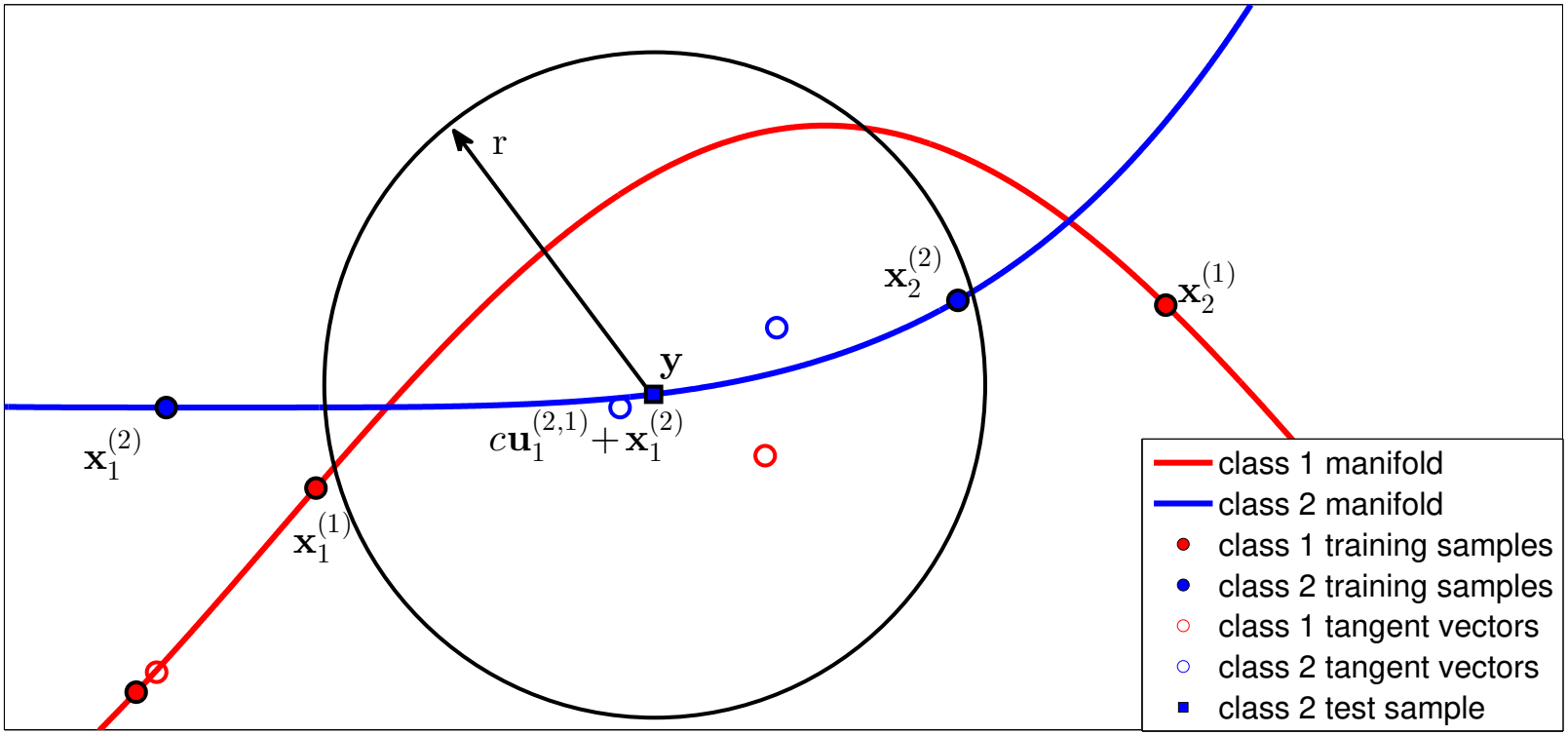
Offline Phase: For each class l, and for each normalized training sample $\boldsymbol{x}_i^{(l)}$ in class l,

1. Find the tangent plane basis vectors $\boldsymbol{u}_1^{(l,i)}, \ldots, \boldsymbol{u}_d^{(l,i)}$ at $\boldsymbol{x}_i^{(l)}$ using local PCA.
2. Save the matrix $\tilde{X}^{(l,i)} := [\boldsymbol{x}_i^{(l)}, c\boldsymbol{u}_1^{(l,i)} + \boldsymbol{x}_i^{(l)}, \ldots, c\boldsymbol{u}_d^{(l,i)} + \boldsymbol{x}_i^{(l)}] \in \mathbb{R}^{m \times (d+1)}$, where $c = r\gamma$, $\gamma \sim \text{unif}(0,1)$, r is the *estimated neighborhood radius*.

Online Phase: For a given normalized test sample $\boldsymbol{y}$,

1. Initialize the dictionary $D_{\boldsymbol{y}} = \emptyset$.
2. For each $\boldsymbol{x}_i^{(l)}$, check to see if $\|\boldsymbol{y} - \boldsymbol{x}_i^{(l)}\|_2 \le r$ or $\|\boldsymbol{y} + \boldsymbol{x}_i^{(l)}\|_2 \le r$. If either is true, add $\tilde{X}^{(l,i)}$ to the dictionary: $D_{\boldsymbol{y}} \leftarrow [D_{\boldsymbol{y}}, \tilde{X}^{(l,i)}]$.
3. Perform SRC, replacing the dictionary X_{tr} with $D_{\boldsymbol{y}}$. Now the dictionary contains only nearby training samples and the (shifted and scaled) basis vectors of their local tangent planes.

Advantage of Including Tangent Vectors



Local PCA and Computational Complexity

Local PCA: PCA on the dataset comprised of the n nearest neighbors of $\boldsymbol{x}_i^{(l)}$. We use the technique of Singer and Wu (2012), which differs from standard local PCA in two ways:

- Samples are shifted by $\boldsymbol{x}_i^{(l)}$ instead of their mean.
- The resulting local data matrix is *weighted* in such a way that samples closer to $\boldsymbol{x}_i^{(l)}$ contribute more.

Computational complexity of LPCA-SRC: with HOMOTOPY (Donoho and Tsaig, 2008) as an ℓ^1 solver, it costs

$$O\Big(\overbrace{m \sum_{l=1}^{L} N_l^2}^{\text{NN-search}} + \overbrace{N_{\text{tr}} m n}^{\text{SVD (local PCA)}} + \overbrace{\frac{N_y}{d} \log\Big(\frac{N_y}{d}\Big)}^{\text{Build dictionary}} + \overbrace{N_y m \kappa + m \kappa^2}^{\text{HOMOTOPY}} \Big),$$

$N_l :=$ # training samples in the lth class;
$N_y :=$ # columns in the pruned dictionary D_y;
$\kappa :=$ # HOMOTOPY iterations.

Local PCA and Computational Complexity

Local PCA: PCA on the dataset comprised of the n nearest neighbors of $\boldsymbol{x}_i^{(l)}$. We use the technique of Singer and Wu (2012), which differs from standard local PCA in two ways:

- Samples are shifted by $\boldsymbol{x}_i^{(l)}$ instead of their mean.
- The resulting local data matrix is *weighted* in such a way that samples closer to $\boldsymbol{x}_i^{(l)}$ contribute more.

Computational complexity of LPCA-SRC: with HOMOTOPY (Donoho and Tsaig, 2008) as an ℓ^1 solver, it costs

$$O\Big(\overbrace{m \sum_{l=1}^{L} N_l^2}^{\text{NN-search}} + \overbrace{N_{\text{tr}} m n}^{\text{SVD (local PCA)}} + \overbrace{\frac{N_{\boldsymbol{y}}}{d} \log\Big(\frac{N_{\boldsymbol{y}}}{d}\Big)}^{\text{Build dictionary}} + \overbrace{N_{\boldsymbol{y}} m \kappa + m\kappa^2}^{\text{HOMOTOPY}} \Big),$$

N_l := # training samples in the lth class;
$N_{\boldsymbol{y}}$:= # columns in the pruned dictionary $D_{\boldsymbol{y}}$;
κ := # HOMOTOPY iterations.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample y to include in D_y.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

  where $r_i^{(l)}$ is the distance between $x_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample y to include in D_y.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

where $r_i^{(l)}$ is the distance between $x_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample $\boldsymbol{y}$ to include in $D_{\boldsymbol{y}}$.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

where $r_i^{(l)}$ is the distance between $\boldsymbol{x}_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample $\boldsymbol{y}$ to include in $D_{\boldsymbol{y}}$.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

where $r_i^{(l)}$ is the distance between $\boldsymbol{x}_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample $\boldsymbol{y}$ to include in $D_{\boldsymbol{y}}$.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

where $r_i^{(l)}$ is the distance between $\boldsymbol{x}_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Setting the Parameters d, n, and r

- We estimate the intrinsic dimension d using cross-validation. It can alternatively be estimated using DANCo (Ceruti, et al., 2014) or Multiscale SVD (Little, et al., 2017), or simply set to $d = 1$.
- We set the number-of-neighbors parameter n using cross-validation. Note that its value must satisfy

$$d \le n < \min_{1 \le l \le L} N_l.$$

- Estimated neighborhood radius r:
  - Recall r is used to check if a training sample is "close enough" to the test sample $\boldsymbol{y}$ to include in $D_{\boldsymbol{y}}$.
  - The local PCA only uses the local class neighborhood of each training sample, i.e., its n-nearest neighbors in the same class.
  - Motivated by preserving this definition of "local class neighborhood," we set

$$r := \operatorname{med}\left\{ r_i^{(l)} \,\middle|\, 1 \le i \le N_l; 1 \le l \le L \right\}$$

where $r_i^{(l)}$ is the distance between $\boldsymbol{x}_i^{(l)}$ and its $(n+1)$st nearest class neighbor.

Outline

Experiments: Methods Compared

- *SRC*: Wright, et al. 2009
- SRC_{pruned}: LPCA-SRC without the the addition of the tangent vectors; equivalently, SRC with the estimated neighborhood radius r
- *Tangent distance classification* (*TDC1* and *TDC2*)
  - Similar to Yang et al.'s *local TDC* algorithm (2012)
  - The test sample is classified to the class with the nearest tangent plane
- *Collaborative representation-based classification with regularized least square* (*CRC-RLS*)
  - Zhang, et al. 2011
  - Essentially SRC with the ℓ^1-norm replaced with the ℓ^2-norm
- *Collaborative neighbor representation classification* (*CNRC*)
  - Waqas, et al. 2012
  - Essentially CRC-RLS with an additional locality constraint
- *Locality sensitive dictionary learning SRC* (*LSDL-SRC*)
  - Wei, et al. 2013
  - Similar to CNRC, but with an additional dictionary learning phase
- *k-nearest neighbor classification* (*kNN*):
  - Test samples are classified to the most-represented class from among their nearest k training samples.
- *k-nearest neighbor over extended dictionary* (*kNN-Ext*):
  - Test samples are classified to the most-represented class from among their nearest k neighbors from the set including the original training samples and the shifted and scaled tangent vectors.

Experiments: Methods Compared

- *SRC*: Wright, et al. 2009
- SRC_{pruned}: LPCA-SRC without the the addition of the tangent vectors; equivalently, SRC with the estimated neighborhood radius r
- *Tangent distance classification* (*TDC1* and *TDC2*)
  - Similar to Yang et al.'s *local TDC* algorithm (2012)
  - The test sample is classified to the class with the nearest tangent plane
- *Collaborative representation-based classification with regularized least square* (*CRC-RLS*)
  - Zhang, et al. 2011
  - Essentially SRC with the ℓ^1-norm replaced with the ℓ^2-norm
- *Collaborative neighbor representation classification* (*CNRC*)
  - Waqas, et al. 2012
  - Essentially CRC-RLS with an additional locality constraint
- *Locality sensitive dictionary learning SRC* (*LSDL-SRC*)
  - Wei, et al. 2013
  - Similar to CNRC, but with an additional dictionary learning phase
- *k-nearest neighbor classification* (*kNN*):
  - Test samples are classified to the most-represented class from among their nearest k training samples.
- *k-nearest neighbor over extended dictionary* (*kNN-Ext*):
  - Test samples are classified to the most-represented class from among their nearest k neighbors from the set including the original training samples and the shifted and scaled tangent vectors.

Experiments: Methods Compared

- *SRC*: Wright, et al. 2009
- SRC_{pruned}: LPCA-SRC without the the addition of the tangent vectors; equivalently, SRC with the estimated neighborhood radius r
- *Tangent distance classification* (*TDC1* and *TDC2*)
  - Similar to Yang et al.'s *local TDC* algorithm (2012)
  - The test sample is classified to the class with the nearest tangent plane
- *Collaborative representation-based classification with regularized least square* (*CRC-RLS*)
  - Zhang, et al. 2011
  - Essentially SRC with the ℓ^1-norm replaced with the ℓ^2-norm
- *Collaborative neighbor representation classification* (*CNRC*)
  - Waqas, et al. 2012
  - Essentially CRC-RLS with an additional locality constraint
- *Locality sensitive dictionary learning SRC* (*LSDL-SRC*)
  - Wei, et al. 2013
  - Similar to CNRC, but with an additional dictionary learning phase
- *k-nearest neighbor classification* (*kNN*):
  - Test samples are classified to the most-represented class from among their nearest k training samples.
- *k-nearest neighbor over extended dictionary* (*kNN-Ext*):
  - Test samples are classified to the most-represented class from among their nearest k neighbors from the set including the original training samples and the shifted and scaled tangent vectors.

Experiments: AR Database

AR face database (Martinez and Benavente, 1998):

- **AR-1:** Contains facial images photographed under varying lighting conditions with varying expression
  - $L = 100,\ N_l = 7,\ m \in \{30, 56\}$ (via PCA on 165×120 pixel images)
- **AR-2:** Additionally contains images with two natural occlusions (sunglasses and scarf)
  - $L = 100,\ N_l = 10,\ m \in \{30, 56\}$ (via PCA on 165×120 pixel images)

(a) Neutral (b) Smile (c) Anger (d) Scream (e) Sunglasses (f) Scarf

Example images (before PCA pre-processing) from the AR database.

Experiments: Extended Yale B Database

Extended Yale B face database (Georghiades, et al. 2001): Contains facial images photographed under varying lighting conditions

- $L = 38,\ N_l = 32,\ m \in \{30, 56\}$ (via PCA on 192×168 pixel images)

Example images (before PCA pre-processing) from Extended Yale B

All training data chosen randomly.
Classification averages over 10 trials.

Classification Results: AR

	$m = 30$						$m = 56$					
	AR-1			AR-2			AR-1			AR-2		
Algorithm	Acc	SE	Time	Acc	SE	Time	Acc	SE	Time	Acc	SE	Time
LPCA-SRC	**0.866**	4.1	7.25	**0.733**	6.0	12.50	**0.954**	2.3	10.53	**0.884**	3.6	35.27
SRC	0.827	4.2	6.11	0.695	4.3	8.87	0.936	2.6	11.39	0.871	2.0	17.67
$\text{SRC}_{\text{pruned}}$	0.828	4.8	3.76	0.709	4.0	5.10	0.935	3.8	11.12	0.878	2.7	16.63
TDC1	0.805	6.5	11.82	0.690	4.1	20.56	0.843	5.1	14.24	0.760	3.7	27.51
TDC2	0.755	19.4	8.89	0.642	16.1	20.93	0.814	11.9	16.79	0.739	3.3	47.57
CRC-RLS	0.753	4.3	0.67	0.597	4.2	0.70	0.922	4.2	1.17	0.824	3.3	1.20
CNRC	0.790	3.9	12.07	0.622	2.5	12.16	0.923	4.1	40.34	0.827	2.8	40.72
LSDL-SRC	0.818	4.0	7.78	0.659	5.6	8.55	0.942	2.0	22.70	0.861	2.2	16.34
kNN	0.525	4.6	0.01	0.410	3.0	0.01	0.630	8.0	0.02	0.430	5.0	0.02
kNN-Ext	0.604	4.5	0.80	0.431	3.7	0.08	0.649	8.2	0.13	0.453	2.9	0.15

Average classification accuracy, standard error ($\times 10^{-3}$), and computation time (in seconds) on AR.

Example: Efficacy of Tangent Vectors

(a) Orig. (b) Recov. (c) TV

(d) Orig. (e) Recov. (f) TV

Examples of tangent vectors containing discriminating information that is lost in the PCA transform. (a), (d): Original images from AR-1 and AR-2; (b), (e): Recovered images from their PCA approximations with $m = 30$; (c), (f): Recovered (shifted and scaled) tangent vectors.

Classification Results: Extended Yale B

	$m = 30$			$m = 56$		
Algorithm	Acc	SE	Time	Acc	SE	Time
LPCA-SRC	**0.905**	2.9	29.20	**0.953**	1.7	72.12
SRC	0.880	2.6	15.58	0.937	2.8	24.70
$\text{SRC}_{\text{pruned}}$	0.880	2.7	15.92	0.937	2.6	23.81
TDC1	0.857	1.0	8.10	0.929	2.0	27.62
TDC2	0.883	3.9	11.68	0.909	2.8	23.51
CRC-RLS	0.732	3.3	0.69	0.882	2.8	0.73
CNRC	0.733	3.3	80.47	0.883	2.9	79.51
LSDL-SRC	0.750	4.8	67.30	0.877	2.5	53.03
kNN	0.430	3.5	0.02	0.535	2.6	0.03
kNN-Ext	0.546	6.9	0.17	0.632	5.6	0.25

Average classification accuracy, standard error ($\times 10^{-3}$), and computation time (in seconds) on Extended Yale B.

Outline

Summary

- Proposed a classification algorithm *LPCA-SRC*, which computes basis vectors of approximate tangent hyperplanes at the training samples in each class and adds them as newly-generated training samples into the training dictionary.
- Each test sample is sparsely decomposed over a dictionary that includes nearby training samples and their corresponding (shifted and scaled) tangent plane basis vectors.
- Its class label is assigned to the class that contributes the most to this decomposition.
- LPCA-SRC regularly achieves higher classification accuracy than SRC and similar methods in cases of *sparse sampling, low noise, small PCA feature dimension, and/or when the class manifolds are close/intersecting*.

Summary

- Proposed a classification algorithm *LPCA-SRC*, which computes basis vectors of approximate tangent hyperplanes at the training samples in each class and adds them as newly-generated training samples into the training dictionary.
- Each test sample is sparsely decomposed over a dictionary that includes nearby training samples and their corresponding (shifted and scaled) tangent plane basis vectors.
- Its class label is assigned to the class that contributes the most to this decomposition.
- LPCA-SRC regularly achieves higher classification accuracy than SRC and similar methods in cases of *sparse sampling, low noise, small PCA feature dimension, and/or when the class manifolds are close/intersecting*.

Summary

- Proposed a classification algorithm *LPCA-SRC*, which computes basis vectors of approximate tangent hyperplanes at the training samples in each class and adds them as newly-generated training samples into the training dictionary.
- Each test sample is sparsely decomposed over a dictionary that includes nearby training samples and their corresponding (shifted and scaled) tangent plane basis vectors.
- Its class label is assigned to the class that contributes the most to this decomposition.
- LPCA-SRC regularly achieves higher classification accuracy than SRC and similar methods in cases of *sparse sampling, low noise, small PCA feature dimension, and/or when the class manifolds are close/intersecting*.

Summary

- Proposed a classification algorithm *LPCA-SRC*, which computes basis vectors of approximate tangent hyperplanes at the training samples in each class and adds them as newly-generated training samples into the training dictionary.
- Each test sample is sparsely decomposed over a dictionary that includes nearby training samples and their corresponding (shifted and scaled) tangent plane basis vectors.
- Its class label is assigned to the class that contributes the most to this decomposition.
- LPCA-SRC regularly achieves higher classification accuracy than SRC and similar methods in cases of *sparse sampling, low noise, small PCA feature dimension, and/or when the class manifolds are close/intersecting*.

Acknowledgments

- N. Saito was partially supported by the ONR grants N00014-12-1-0177, N00014-16-1-2255, and the NSF grant DMS-1418779.
- C. Weaver's research on this project was conducted with government support under contract FA9550-11-C-0028 and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. She was also supported by NSF VIGRE DMS-0636297 and NSF DMS-1418779.