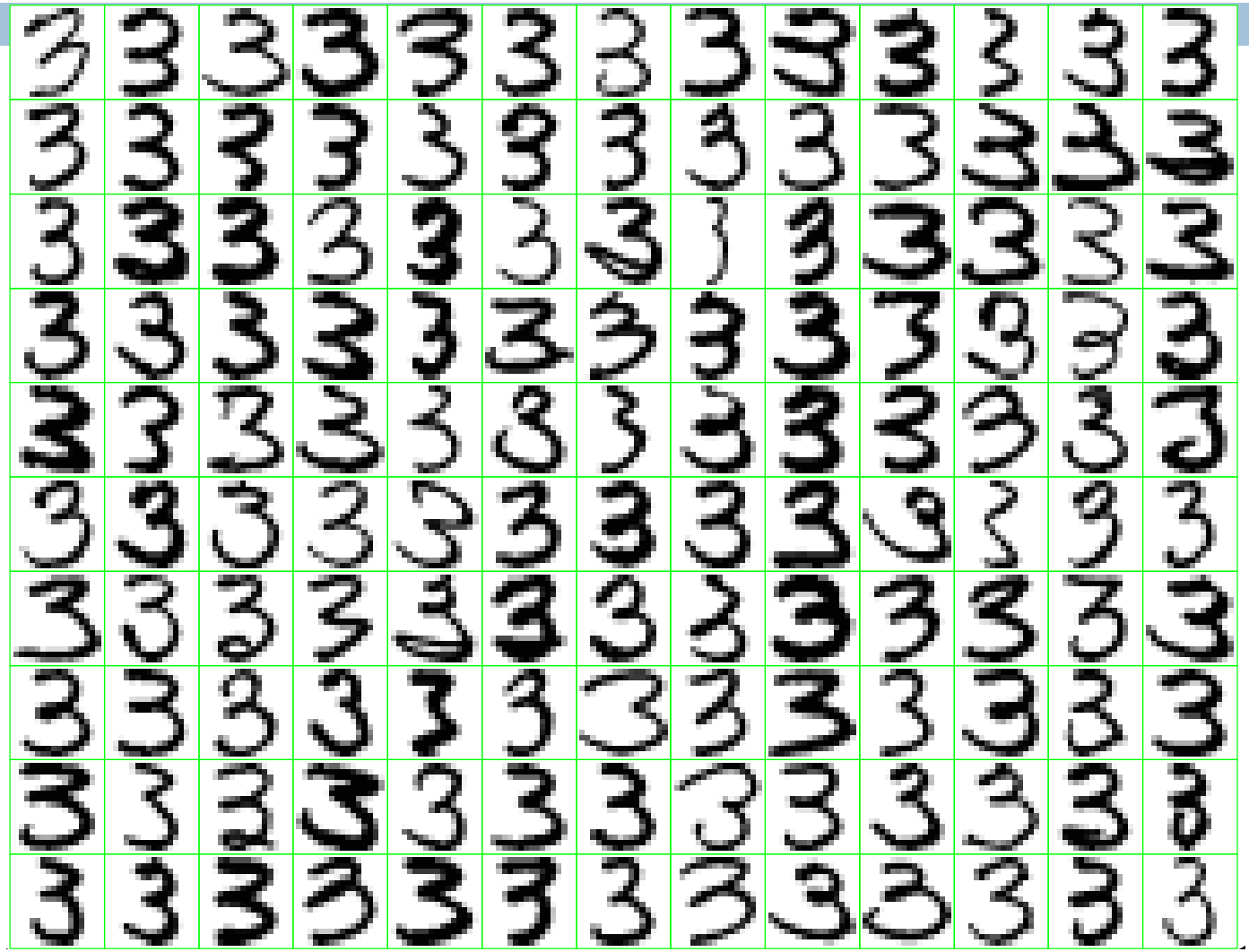
Lecture 8: Principal Component Analysis; Kernel PCA

Lester Mackey

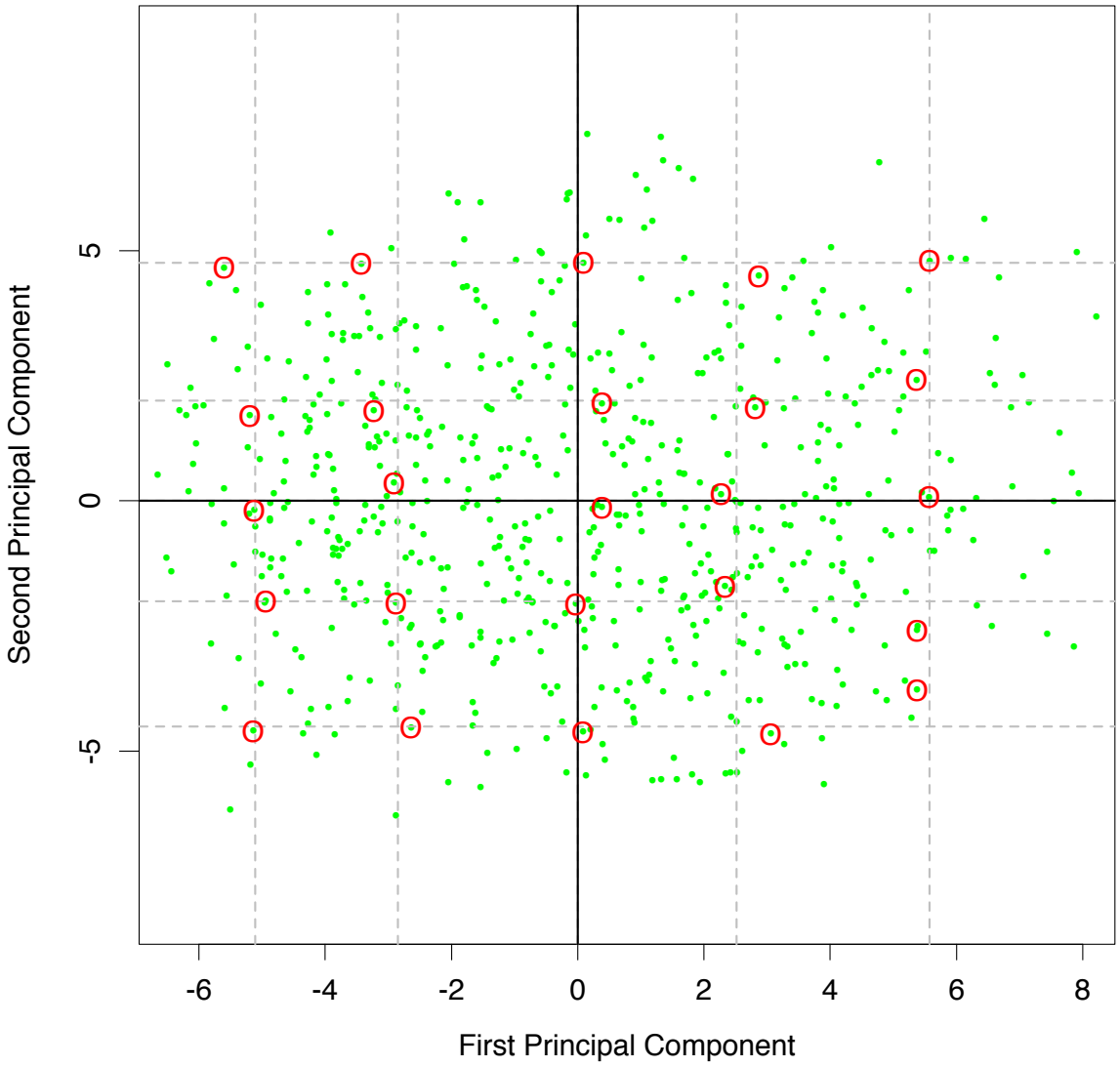
April 23, 2014

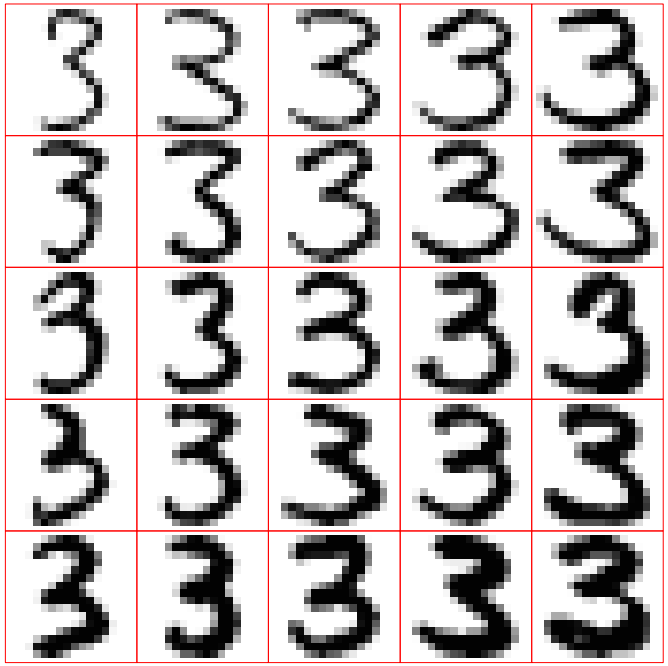
PCA example: digit data



130 threes, a subset of 638 such threes and part of the handwritten digit dataset. Each three is a 16×16 greyscale image, and the variables X_j, $j = 1, \ldots, 256$ are the greyscale values for each pixel.

PCA example: digit data



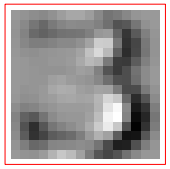


PCA example: digit data

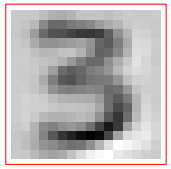
Two-component model has the form

$$\begin{aligned} \hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \end{aligned}$$

$+ \lambda_1 \cdot$



$+ \lambda_2 \cdot$



.

Here we have displayed the first two principal component directions, v_1 and v_2, as images.

PCA in the wild: Eigen-faces

Courtesy: Percy Liang

- Turk and Pentland, 1991

  - $d =$ number of pixels
  - Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
  - $\mathbf{x}_{ji} =$ intensity of the j-th pixel in image i

$$\mathbf{X}_{d\times n} \approx \mathbf{U}_{d\times k} \mathbf{Z}_{k\times n}$$

$$\left(\begin{array}{ccc} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{array} \right) \approx \left(\begin{array}{ccc} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_k \\ | & & | \end{array} \right) \left(\begin{array}{ccc} | & & | \\ \mathbf{z}_1 & \cdots & \mathbf{z}_n \\ | & & | \end{array} \right)$$

Idea: $\mathbf{z}_i$ more "meaningful" representation of i-th face than $\mathbf{x}_i$

Can use $\mathbf{z}_i$ for nearest-neighbor classification

Much faster: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \gg k$

PCA in the wild: Latent semantic analysis

Courtesy: Percy Liang

- Deerwester/Dumais/Harshman, 1990
  - $d =$ number of words in the vocabulary
  - Each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of word counts
  - $\mathbf{x}_{ji} =$ frequency of word j in document i

$$\underset{\mathbf{X}_{d\times n}}{\begin{pmatrix} \text{stocks: } 2 & \cdots & 0 \\ \text{chairman: } 4 & \cdots & 1 \\ \text{the: } 8 & \cdots & 7 \\ \cdots & \vdots & \vdots \\ \text{wins: } 0 & \cdots & 2 \\ \text{game: } 1 & \cdots & 3 \end{pmatrix}} \approx \underset{\mathbf{U}_{d\times k}}{\begin{pmatrix} 0.4 & \cdots & -0.001 \\ 0.8 & \cdots & 0.03 \\ 0.01 & \cdots & 0.04 \\ \vdots & \cdots & \vdots \\ 0.002 & \cdots & 2.3 \\ 0.003 & \cdots & 1.9 \end{pmatrix}} \underset{\mathbf{Z}_{k\times n}}{\begin{pmatrix} | & & | \\ \mathbf{z}_1 & \cdots & \mathbf{z}_n \\ | & & | \end{pmatrix}}$$

How to measure similarity between two documents?

$\mathbf{z}_1^\top \mathbf{z}_2$ is probably better than $\mathbf{x}_1^\top \mathbf{x}_2$

Applications: information retrieval

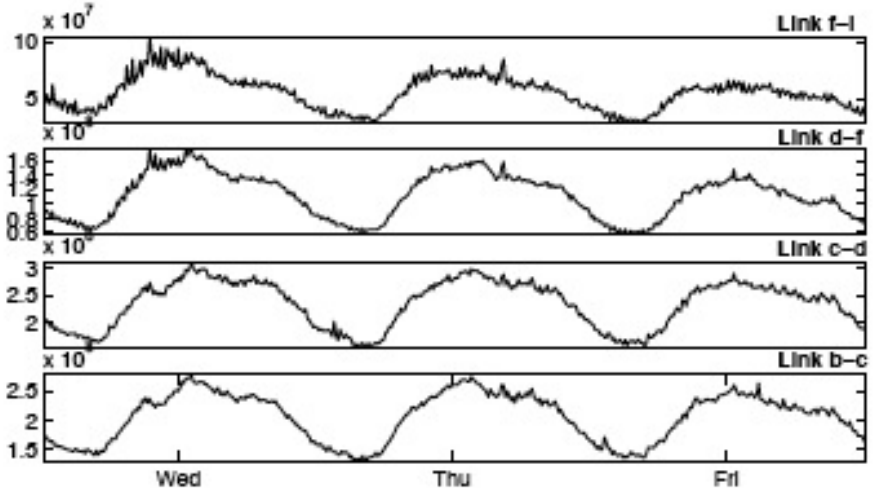
Note: no computational savings; original $\mathbf{x}$ is already sparse

PCA in the wild: Anomaly detection

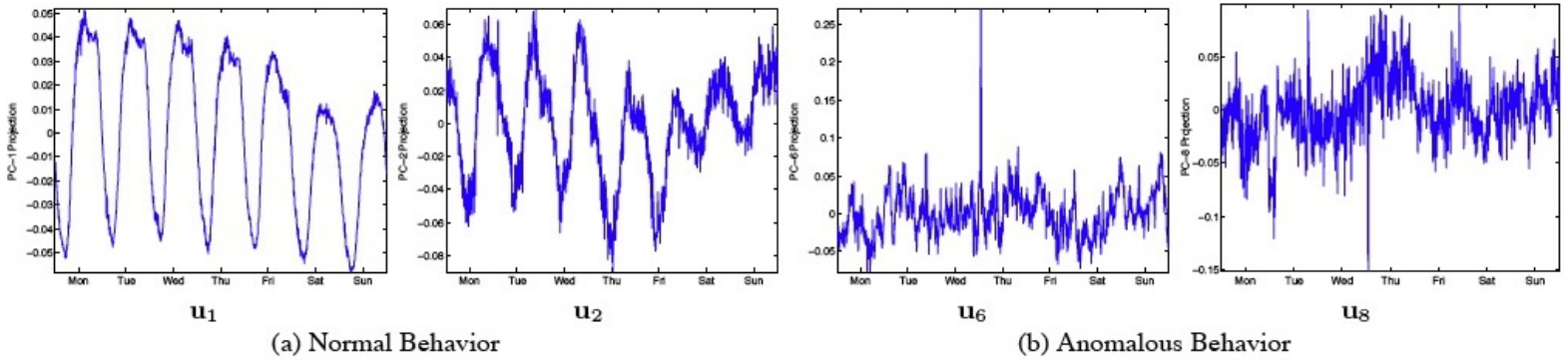
Courtesy: Percy Liang

- Lakhina/Crovella/Diot, '04

$\mathbf{x}_{ji}$ = amount of traffic on link j in the network during each time interval i



Model assumption: total traffic is sum of flows along a few "paths"
Apply PCA: each principal component intuitively represents a "path"
Anomaly when traffic deviates from first few principal components



PCA in the wild: Part-of-speech tagging

Courtesy: Percy Liang

- Schütze, '95

  Part-of-speech (POS) tagging task:

Input:	I	like	reducing	the	dimensionality	of	data	.
Output:	NOUN	VERB	VERB(-ING)	DET	NOUN	PREP	NOUN	.

  Each $\mathbf{x}_i$ is (the context distribution of) a word.

  $\mathbf{x}_{ji}$ is number of times word i appeared in context j

  Key idea: words appearing in similar contexts tend to have the same POS tags; so cluster using the contexts of each word type

  Problem: contexts are too sparse

  Solution: run PCA first, then cluster using new representation

PCA in the wild: Multi-task learning

Courtesy: Percy Liang

- Ando & Zhang 05

- Have n related tasks (classify documents for various users)
- Each task has a linear classifier with weights $\mathbf{x}_i$
- Want to share structure between classifiers

One step of their procedure:

given n linear classifiers $\mathbf{x}_1, \ldots, \mathbf{x}_n$,
run PCA to identify shared structure:

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \ldots & \mathbf{x}_n \\ | & & | \end{pmatrix} \approxeq \mathbf{U}\mathbf{Z}$$

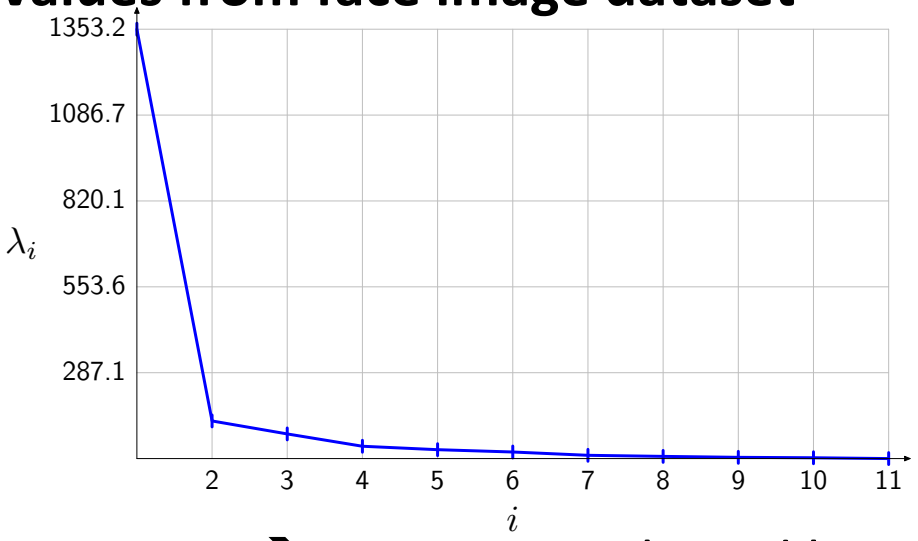
Each column of U is an eigen-classifier

Other step of their procedure:

Retrain classifiers, regularizing towards subspace $\mathbf{U}$

Choosing a number of components

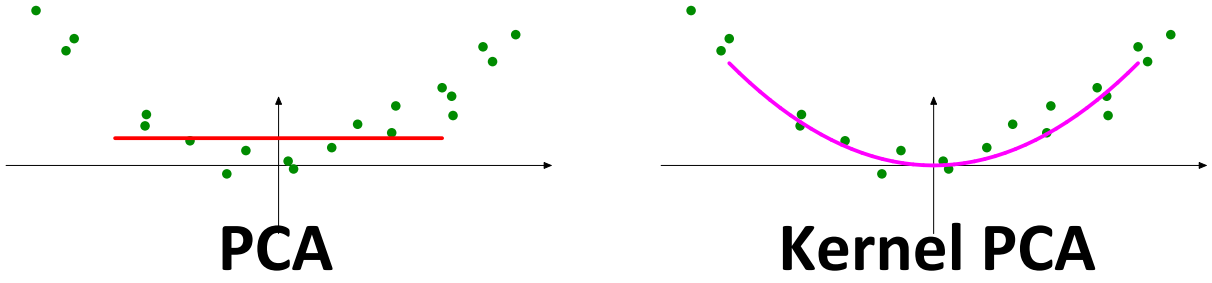
- As in the clustering setting, an important problem with no single solution
  - May be constrained by goals (visualization), resources, or minimum fraction of variance to be explained
  - **Note**: Eigenvalue magnitudes determine explained variance
    - e.g., **Eigenvalues from face image dataset**



    - Rapid decay to zero ➔ variance explained by a few components
    - Could look for elbow or compare with reference distribution

PCA limitations and extensions

- Squared Euclidean reconstruction error not appropriate for all data types
  - Various extensions, like **exponential family PCA**, have been developed for binary, categorical, count, and nonnegative data (e.g., Collins/Dasgupta/Schapire, A Generalization of Principal Component Analysis to the Exponential Family)
- PCA can only find linear compressions of data
  - What if data best summarized in a non-linear fashion?
  - **Kernel PCA** allows us to perform such non-linear dimensionality reduction



Credit: Percy Liang

Blackboard discussion

- See lecture notes