CO902
Probabilistic and statistical inference

Lecture 6

Tom Nichols
Department of Statistics &
Warwick Manufacturing Group

t.e.nichols@warwick.ac.uk

Admin

- Project (“Written assignment”)
  - Due Wednesday 11 Feb at noon (or today, if you want! ☺)
  - Questions!?
- Presentation (“Critical Reading Assignment”)
  - … at end of lecture today

Critical Reading Assignment

- 10 minute presentation
- Based on journal paper about or using Machine Learning methods
  Some suggestions on the webpage; other's OK, but contact me
  Must notify me of your article by 18 Feb (next Mon!)
- Aim for ~5 slides, not much more
- Try to think of intuitive descriptions for any algorithms/procedures used
- Only 10 minutes! Not a lecture or a tutorial
  Can't possibly explain everything…
  But should be able to express general ideas in play
- Don't read your slides!
  Slides shouldn't have full sentences,
  Just key words/phrase
  to anchor audience's attention, and
  to help guide/remind you of the flow
- Make good use of pretty, meaningful pictures when possible
- Look at the audience as much as possible!
- Practice!

Outline of course

A. Basics: Probability, random variables (RVs), common distributions, introduction to statistical inference

B. Supervised learning: Regression, classification, including high-dimensional issues and Bayesian approaches

C. Unsupervised learning: Dimensionality reduction, clustering and mixture models

D. Networks: Probabilistic graphical models, learning in graphical models, inferring network structure

Supervised Learning Redux (1)

- Given samples $\{\mathbf{X}_i, Y_i\},\ i = 1..n$ build tool to predict Y_{new} for a new case using only $\mathbf{X}_{\text{new}}$

Binary inputs, binary output

$$\mathbf{X}_i \in \{0,1\}^d,\ Y_i \in \{0,1\}$$

Cont. inputs, discrete output

$$\mathbf{X}_i \in \mathbb{R}^d,\ Y_i \in \{1,2,\ldots k\}$$

- **Classification:** Binary/discrete output
  Optimal classifier based on…
  - Discrete input

$$P(Y = k|\mathbf{X} = \mathbf{x}) \propto P(\mathbf{X} = \mathbf{x}|Y = k)P(Y = k)$$

  - Continuous input

$$P(Y = k|\mathbf{X} = \mathbf{x}) \propto p(\mathbf{x}|Y = k)P(Y = k)$$

- **Class conditional distribution**
  "Generative" model for data from class k

e.g. for continuous input… $p_k(\mathbf{x}) = p(\mathbf{x}|Y = k)$

- **Estimation**
  Class conditional must estimated, typically with parameterized distribution

e.g. for continuous input… $\hat{p}_k(\mathbf{x}) = p(\mathbf{x}|Y = k, \hat{\theta}_k)$

e.g. Bernoulli success rates; mean and variance of a Gaussian; etc

Supervised Learning Redux (2)

- **Linear Discriminant Analysis** (linear decision boundary)
  Gaussian generative model, equal covariance Σ over all classes k
- **Quadratic Discriminant Analysis** (curved decision boundary)
  Gaussian generative model, class-specific covariance Σ_k
- **Naïve Bayes classifier**
  Based on independence over d input dimensions, using

e.g. for continuous input… $$p_k(\mathbf{x}) = \prod_{j=1}^{d} p(x_j|Y=k)$$

- **Cross Validation**
  Attempt to estimate classifier accuracy with unseen data
- **k-fold Cross-Validation**
  Run classifier k times, each time using (k-1)/k × N samples
- **Leave One Out Cross-Validation** (LOOCV)
  Run classifier *N* times, each time using *N*–1 samples
  Has least biased estimate of true error, but more variable estimate than k-fold
  More computationally intensive

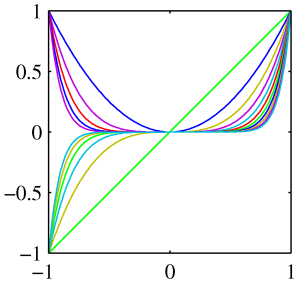
Supervised Learning Redux (3)

Cont. inputs, cont. output
$\mathbf{X}_i \in \mathbb{R}^d,\ Y_i \in \mathbb{R}$

- **Least Squares Regression**
  Minimize sum of squared errors between observed and predicted response
  Or, maximize likelihood of *iid* Gaussian errors in prediction

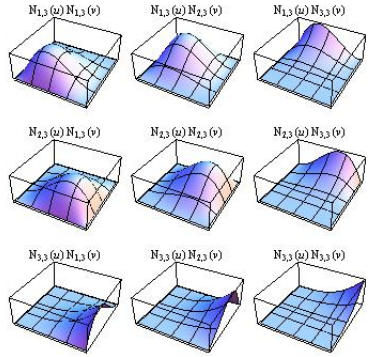
$$\hat{Y}(\mathbf{X}, \mathbf{w}) = \mathbf{w}^\top \mathbf{X} \qquad \hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^{-} \mathbf{Y}$$

- **Polynomial Regression**
  Uses polynomial expansion of inputs, to get more flexibility



$$\phi(X) \quad = \quad [1\ X\ X^2 \ldots X^k]^T \qquad \hat{\mathbf{w}} = \mathbf{\Phi}^{-} \mathbf{Y}$$

- **Arbitrary Basis Sets**
  E.g. splines, wavelets, Gaussians, etc.
  More compact support than polynomial basis



$$\phi(\mathbf{X}) \quad = \quad [1\ \phi_1(\mathbf{X}) \ldots \phi_k(\mathbf{X})]^T \qquad \hat{\mathbf{w}} = \mathbf{\Phi}^{-} \mathbf{Y}$$

- **Ridge regression**
  Penalized Maximum Likelihood **Or** Bayesian MAP solution

$$\hat{\mathbf{w}} \quad = \quad (\mathbf{\Phi}^T \mathbf{\Phi} + \lambda \mathbf{I}_k)^{-1} \mathbf{\Phi}^T \mathbf{Y}$$

  Regularizes fit when too many (or just redundant) parameters

Outline of course

A. Basics: Probability, random variables (RVs), common distributions, introduction to statistical inference

B. Supervised learning: Regression, classification, including high-dimensional issues and Bayesian approaches

C. Unsupervised learning: Dimensionality reduction, clustering and mixture models

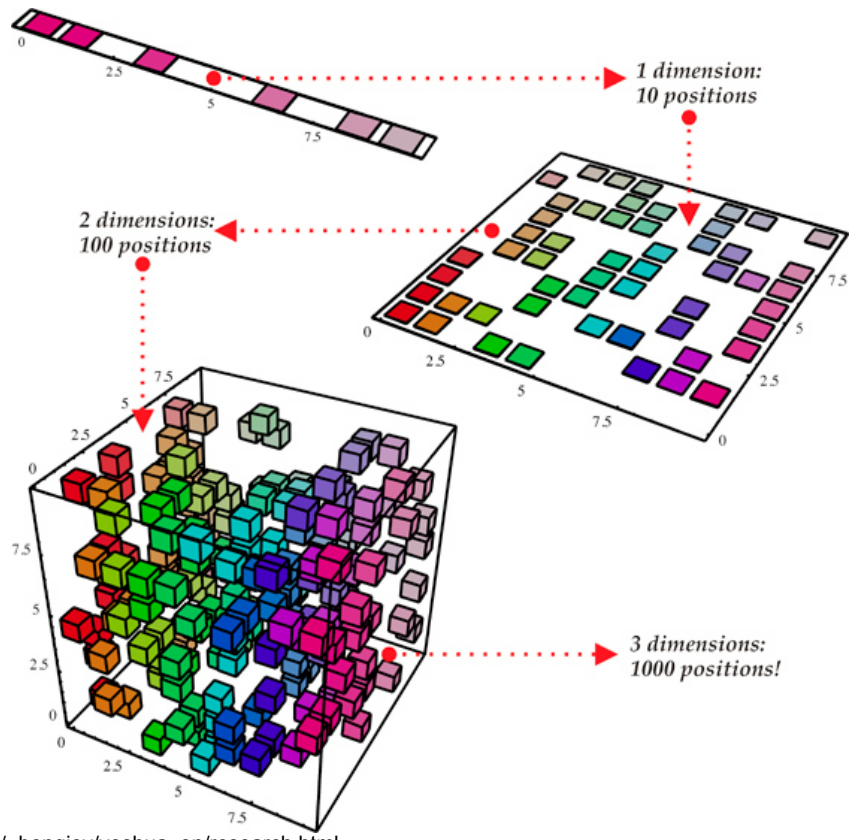
D. Networks: Probabilistic graphical models, learning in graphical models, inferring network structure

Unsupervised learning

- Unsupervised learning:
  - Finding "intrinsic structure" in data
  - Or, finding patterns without knowing what you're looking for
- Two key classes of unsupervised learning:
  - Dimensionality reduction
  - Clustering
- **Un**supervised: you **don't** start with a "labelled" dataset

Curse of dimensionality

- Data in high dimensions can be troublesome
- High-dimensional data pose several problems:
  - Statistical inference is very difficult, we've seen this over and over!
  - Computational issues can also become a problem
- Gives rise to phrase **curse of dimensionality** (phrase due to Bellman, ~1960)



Consider following exercise:

- Fill $[0,1]^D$ space with sufficient data points such that,
- For each location $x_0 \in [0,1]^D$, a fixed number of data points are within a distance δ (on average)
- Let D grow… required number of observations will grow exponentially!

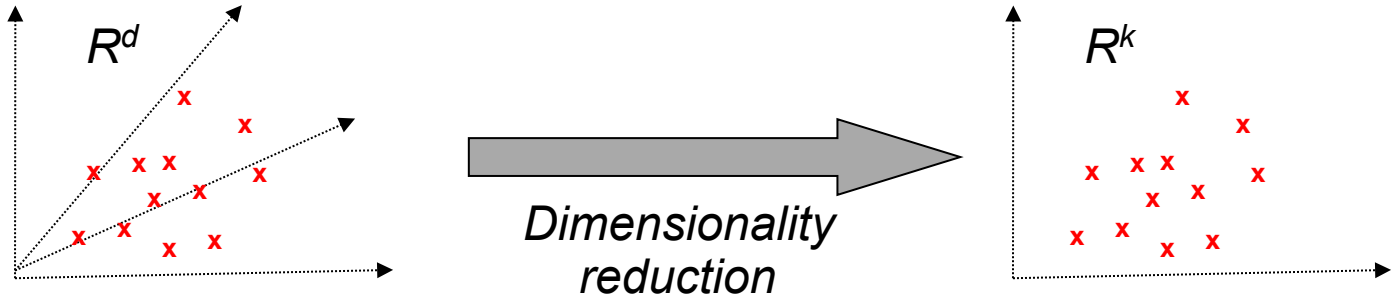
In other words

- For typical (finite) n, D-dim space is sparsely populated

http://www.iro.umontreal.ca/~bengioy/yoshua_en/research.html

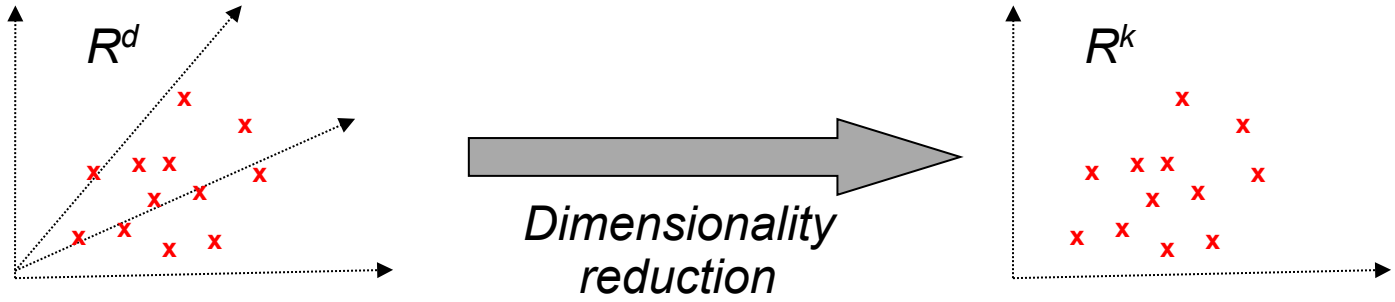
Dimensionality reduction

- **Dimensionality reduction**: unsupervised learning problem in which goal is to obtain a lower-dimensional representation of the data losing as little useful information as possible



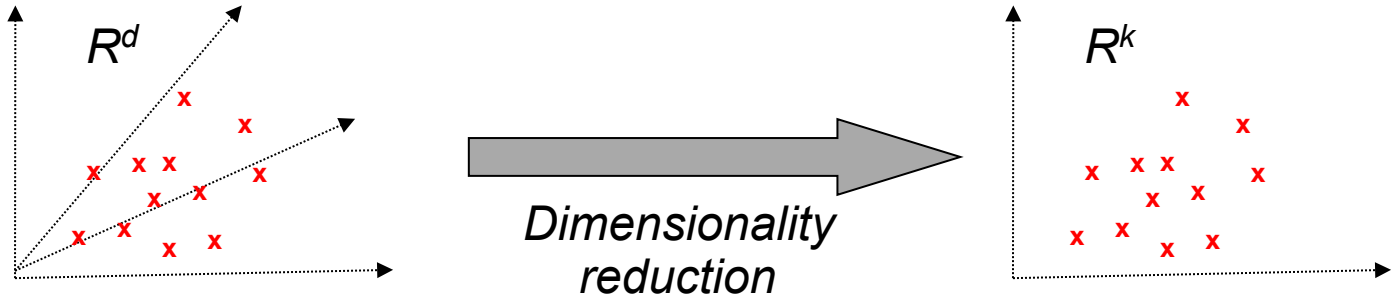
- Typically *k<<d*
- Two reasons to do this:
  - **Pre-processing**: Simplify data for subsequent analysis
  - **Visualization**: *If k<=3,* data can be looked at

Dimensionality reduction



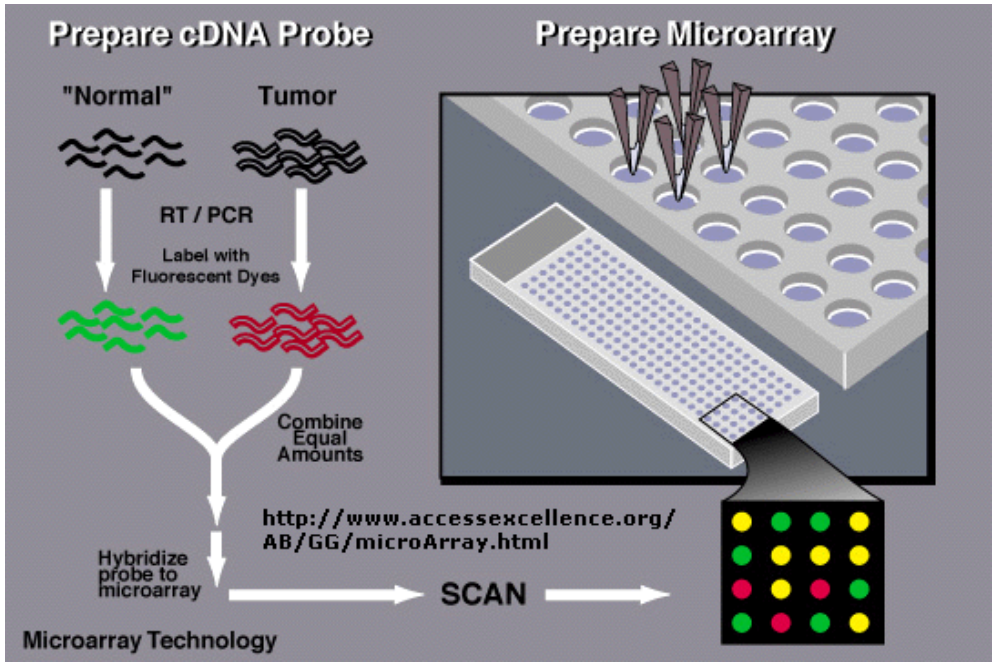
- The low-dimensional data are co-ordinates in a space whose (few) axes are somehow constructed from the original data
- These axes capture (a small number of) important modes of variation
- **Is this just *variable selection ? No!***
- Again, something we do a *lot* of in making sense of a complex world:
  - Mental and verbal descriptions of people in terms of small number of characteristics
  - "Left" and "right" in politics: $R^{1000} \rightarrow R^1$!

Dimensionality reduction



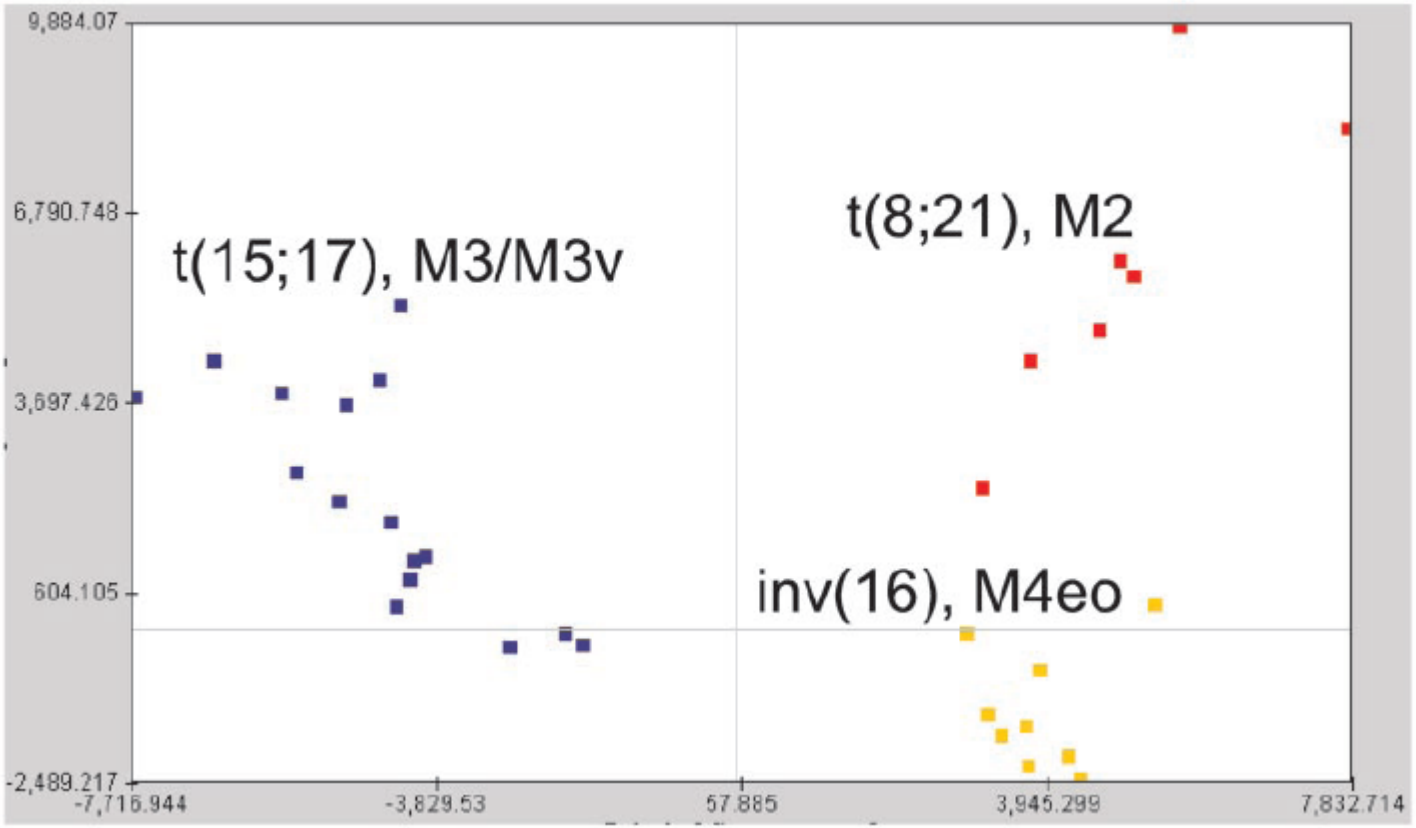
- Example:
  - Suppose you described people by **height**, **weight** and **IQ**
  - Despite fitness/obesity, height & weight are strongly related
    - Roughly, short folks weight less than tall folks
  - Try to replace height & weight with "size"
    - If height & weight exactly linearly related, you haven't lost anything in going from 3 to 2 numbers
  - We've "compressed" the data by removing redundancy
  - Made statistical tasks like density estimation etc. easier

Gene expression microarrays



- Roughly speaking, gene expression is the "activity level" of a gene
- Microarrays can measure all 30,000 genes in one go!
- That is, you get a vector in R^{30k} under each condition, or across a range of conditions, through time etc...
- Now widely used in all areas of biomedical discovery, e.g. cancer

Dimensionality reduction



Schoch et al. (2002), *PNAS* 99(15):10008-10013

- Schoch et al., PNAS 2002 on *Acute Myeloid Leukemia*
  - Gene expression (1000 genes) on bone marrow (32 patients)
  - Simple dimensionality reduction revealed clinically distinct sub-types
    - From 32 × 1000 matrix…. to a 32 × 2 matrix
      These 2 dimensions capture most (67%) of the variability!

Linear projections

- Simplest way to reduce dimensionality is to project the data linearly:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{U}_1^T \mathbf{X}_i \\ \mathbf{U}_1 &: d \times k \end{aligned}$$

$$\mathbf{X}_1 \ldots \mathbf{X}_n$$

$$\mathbf{X}_i \in \mathbb{R}^d$$

- Here, the new axes are simply the columns of **projection matrix** ***U***
- The low-dimensional data ***Y*** are new co-ordinates for the space spanned by columns of ***U*** (the column space of ***U***)
- A linear projection called **principal components analysis** **or** **PCA** is very widely used and will be our focus today
- Let's start with k=1…

PCA in one dimension

- **Linear projection,** k=1
- Think back to height-weight example, we only really care about *direction* we're projecting onto, length just results in a scale factor for the final projections.
- Simplest to assume **u** unit length**, u' u = 1**

- *What* do we want to maximize?
- The (sample) **variance in the projected space**
  - One way of capturing the informativeness of the projection
    - a projection onto a point squashes away all the information, while a "well spread out" projection is good
- Let's choose **u** so as to maximise variance

- But first let's review eigenvalues/eigenvectors…

PCA in one dimension

- Want $\mathbf{u}$ that maximises $\quad \widehat{\mathrm{Var}}(\mathbf{Y}) = \mathbf{u}'\mathbf{S}\mathbf{u} \quad \text{s.t.} \quad \mathbf{u}'\mathbf{u} = 1$

  $d \times d$ sample covariance: $\quad \mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})'$

- Solution must satisfy $\quad \mathbf{S}\mathbf{u} = \lambda\mathbf{u} \quad \text{s.t.} \quad \mathbf{u}'\mathbf{u} = 1$

  But, this solution is not unique; every eigenvector/value of $\mathbf{S}$!

- Want solution that maximises

  $$\widehat{\mathrm{Var}}(\mathbf{Y}) = \mathbf{u}'\mathbf{S}\mathbf{u} = \mathbf{u}'(\lambda\mathbf{u}) = \lambda$$

  That is, the eigenvector with the largest eigenvalue!

- This is the **first principal component** of the data
- Interestingly, it's also the best reconstruction in a least squares sense!

PCA in general

- For the general case k>1, we need to write down the variance of Y's in k-dimensions

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{U}_1^T \mathbf{X}_i \\ \mathbf{U}_1 &: d \times k \end{aligned}$$

  - Not the $k \times k$ covariance, but
  - The average squared distance to the mean, i.e.
  - The (sample) average squared L2 norm of $\mathbf{Y}$ "centered":

$$\begin{aligned} VAR(\mathbf{Y}) &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Y}_i - \bar{\mathbf{Y}})^T(\mathbf{Y}_i - \bar{\mathbf{Y}}) \\ \bar{\mathbf{Y}} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{Y}_i \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{U}_1^T\mathbf{X}_i = \mathbf{U}_1^T\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i\right) = \mathbf{U}_1^T\bar{\mathbf{X}} \end{aligned}$$

PCA in general

- **Maximand is the variance in projected space**
  (ps**:** full derivation of PCA *not* on exam, only key results and intuition)

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{U}_1^T \mathbf{X}_i \\ \mathbf{U}_1 &: d \times k \end{aligned}$$

$$\begin{aligned} VAR(\mathbf{Y}) &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Y}_i - \bar{\mathbf{Y}})^T(\mathbf{Y}_i - \bar{\mathbf{Y}}) \\ \bar{\mathbf{Y}} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{Y}_i \end{aligned}$$

- Useful "trace" trick:

$$\begin{aligned} (\mathbf{Y}_i - \bar{\mathbf{Y}})^T(\mathbf{Y}_i - \bar{\mathbf{Y}}) &= [\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}})]^T[\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}})] \\ &= (\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_1\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}}) \\ &= \mathrm{Tr}[(\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_1\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}})] \\ &= \mathrm{Tr}[\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_1] \end{aligned}$$

(cyclic property of the trace)

Objective function

- **This gives:**

$$
\begin{aligned}
VAR(\mathbf{Y}) &= \frac{1}{n}\sum_{i=1}^{n} \mathrm{Tr}[\mathbf{U}_1^T(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_1] \\
&= \mathrm{Tr}[\mathbf{U}_1^T\mathbf{S}\mathbf{U}_1] \\
\mathbf{S} &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T
\end{aligned}
$$

- Would like to maximize this quantity. But this won't make sense unless we constrain $\mathbf{U_1}$. We want it's columns to be unit length, so use a Lagrange multiplier:

$$
\begin{aligned}
J(\mathbf{U}_1) &= \mathrm{Tr}[\mathbf{U}_1^T\mathbf{S}\mathbf{U}_1] + \sum_{j=1}^{k}\lambda_j(1 - \mathbf{u}_j^T\mathbf{u}_j) \\
\mathbf{U}_1 &= [\mathbf{u}_1 \ldots \mathbf{u}_k]
\end{aligned}
$$

Maximization

- Maximise

$$J(\mathbf{U}_1) = \mathrm{Tr}[\mathbf{U}_1^T \mathbf{S} \mathbf{U}_1] + \sum_{j=1}^{k} \lambda_j (1 - \mathbf{u}_j^T \mathbf{u}_j)$$

- Now:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\mathbf{U}_1} \lambda_j (1 - \mathbf{u}_j^T \mathbf{u}_j) &= [0 \ldots -2\lambda_j \mathbf{u}_j \ldots 0]^T \\
\frac{\mathrm{d}}{\mathrm{d}\mathbf{U}_1} \sum_{j=1}^{k} \lambda_j (1 - \mathbf{u}_j^T \mathbf{u}_j) &= [-2\lambda_1 \mathbf{u}_1 \ldots -2\lambda_k \mathbf{u}_k]^T \\
&= -2(\mathbf{U}_1 \mathbf{\Lambda})^T \\
\textit{where…} \quad \mathbf{\Lambda} &= \mathrm{diag}([\lambda_1 \ldots \lambda_k])
\end{aligned}$$

Maximization

- Also:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{X}}\mathrm{Tr}[\mathbf{X}^T\mathbf{A}\mathbf{X}] = ((\mathbf{A}+\mathbf{A}^T)\mathbf{X})^T$$ *(Magnus & Neudecker, p178)*

- Setting derivative of *J* wrt $\mathbf{U_1}$ to zero:

$$\mathbf{S}\mathbf{U}_1 = \mathbf{U}_1\mathbf{\Lambda}$$

- The full eigen-decomposition
- With *k=d*, it's the *spectral decomposition*
- *In other words: the columns of U_1 are simply k eigenvectors of the sample covariance matrix S*
- But *which k* ?

Maximum variance solution

- Solution:

$$\mathbf{S}\mathbf{U}_1 = \mathbf{U}_1\mathbf{\Lambda}$$

- Overall variance:

$$\begin{aligned} VAR(\mathbf{Y}) &= \mathrm{Tr}[\mathbf{U}_1^T\mathbf{S}\mathbf{U}_1] \\ &= \mathrm{Tr}[\mathbf{U}_1^T\mathbf{U}_1\Lambda] \\ &= \lambda_1 + \ldots + \lambda_k \end{aligned}$$

- Solution: pick the *k* eigenvectors corresponding to the k largest eigenvalues.

- This is a nice, simple solution, can be computed easily using standard matrix operations

- Crucial! Tells us that the (sample) variance explained by k-dimensional approximate is sum of k largest eigenvalues!

PCA as a transformation: k=d case

- What happens when *k=d*?
- **Q: What's the sample covariance matrix of the projected data Y?**

PCA as a transformation: k=d case

- What happens when *k=d*?
- **Q: What's the sample covariance matrix of the projected data Y?**

- It's diagonal!
- Implication: we can *always* make the data uncorrelated, simply by rotating so the variances lie "along the axes"...

PCA: a second view

- Remarkably, PCA is also the best low-dimensional reconstruction from the squared error point of view:

$\{ \mathbf{u}_j \}$ an *arbitrary* orthonormal basis set…

$$
\begin{aligned}
\mathbf{X}_i &= \sum_{j=1}^{d} \mathbf{u}_j \alpha_j \\
&= \sum_{j=1}^{d} \mathbf{u}_j (\mathbf{u}_j^T \mathbf{X}_i) \\
&= \mathbf{U}\mathbf{U}^T \mathbf{X}_i \\
\mathbf{X}_i &= \mathbf{U}_1 \mathbf{U}_1^T \mathbf{X}_i + \mathbf{U}_2 \mathbf{U}_2^T \mathbf{X}_i \\
\mathbf{U} &= [\mathbf{u}_1 \dots \mathbf{u}_d] \\
\mathbf{U}^T\mathbf{U} &= \mathbf{I}_d \\
\mathbf{U} &= [\mathbf{U}_1 \; \mathbf{U}_2] \\
\mathbf{U}_1 &: d \times k \\
\mathbf{U}_2 &: d \times (d-k)
\end{aligned}
$$

exact fit found with these α's

… s.t. that we want first k to approximate $\mathbf{X}_i$

See also Bishop (PRML), §12.1.2

PCA: a second view

- Low-dimensional approximation:

$$\hat{\mathbf{X}}_i = \mathbf{U}_1\mathbf{U}_1^T\mathbf{X}_i + \mathbf{U}_2\mathbf{b}$$

per obs. fit w/ $\mathbf{U}_1$ — approx. w/ $\mathbf{U}_2$

- Reconstruction error:

$$\begin{aligned}\mathbf{X}_i - \hat{\mathbf{X}}_i &= \mathbf{U}_2\mathbf{U}_2^T\mathbf{X}_i - \mathbf{U}_2\mathbf{b} \\ &= \mathbf{U}_2(\mathbf{U}_2^T\mathbf{X}_i - \mathbf{b})\end{aligned}$$

- Reconstruction error sum of squares (over d dim's), summed (over n obs)

$$\sum_{i=1}^{n} \|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2$$

- Derivative:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\mathbf{b}}\|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2 &= -2\mathbf{U}_2^T\mathbf{U}_2(\mathbf{U}_2^T\mathbf{X}_i - \mathbf{b}) \\ &= -2(\mathbf{U}_2^T\mathbf{X}_i - \mathbf{b})\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{b}}\sum_{i=1}^{n}\|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2 = -2\sum_{i=1}^{n}(\mathbf{U}_2^T\mathbf{X}_i - \mathbf{b})$$

PCA: a second view

- Setting to zero, solving for **b**:

$$\mathbf{b} = \mathbf{U}_2^T \bar{\mathbf{X}}$$

- This gives:

$$\begin{aligned} \|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2 &= \|\mathbf{U}_2\mathbf{U}_2^T(\mathbf{X}_i - \bar{\mathbf{X}})\|^2 \\ &= (\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_2\mathbf{U}_2^T(\mathbf{X}_i - \bar{\mathbf{X}}) \\ &= \mathrm{tr}[(\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_2\mathbf{U}_2^T(\mathbf{X}_i - \bar{\mathbf{X}})] \\ &= \mathrm{tr}[\mathbf{U}_2^T(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T\mathbf{U}_2] \end{aligned}$$

$\mathbf{U}_2^T\mathbf{U}_2$ this is identity

$$\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2 = \mathrm{tr}[\mathbf{U}_2^T\mathbf{S}\mathbf{U}_2]$$

$$\mathbf{S} = \frac{1}{n}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T$$

PCA: a second view

- Using a Lagrange multiplier as before:

$$\begin{aligned} J(\mathbf{U}_2) &= \mathrm{tr}[\mathbf{U}_2^T \mathbf{S} \mathbf{U}_2] + \sum_{j=1}^{k} \lambda_j (1 - \mathbf{u}_j^T \mathbf{u}_j) \\ \mathbf{U}_2 &= [\mathbf{u}_1 \ldots \mathbf{u}_k] \end{aligned}$$

- Setting derivative to zero and solving yields:

$$\mathbf{S}\mathbf{U}_2 = \mathbf{U}_2 \mathbf{\Lambda}$$

- Overall error is:

$$\begin{aligned} \mathrm{tr}[\mathbf{U}_2^T \mathbf{S} \mathbf{U}_2] &= \mathrm{tr}[\mathbf{U}_2^T \mathbf{U}_2 \mathbf{\Lambda}] \\ &= \lambda_{k+1} + \ldots + \lambda_d \end{aligned}$$

- Notice subscripts: we want the *smallest* error, so solution is same as before!

PCA: Approximation in Anger

- What about actual approximations?
  - We have this expression, but it depends on all n eigenvectors

$$\hat{\mathbf{X}}_i \;=\; \mathbf{U}_1\mathbf{U}_1^T\mathbf{X}_i + \mathbf{U}_2\mathbf{U}_2^T\bar{\mathbf{X}}$$

  - Some algebra shows that you only need the first k… (PRML, §12.1.3)

$$\hat{\mathbf{X}}_i = \bar{\mathbf{X}} + \sum_{j=1}^{k}(\mathbf{X}_i^T\mathbf{u}_j - \bar{\mathbf{X}}^T\mathbf{u}_j)\mathbf{u}_j$$

    and, if data are centred this is just

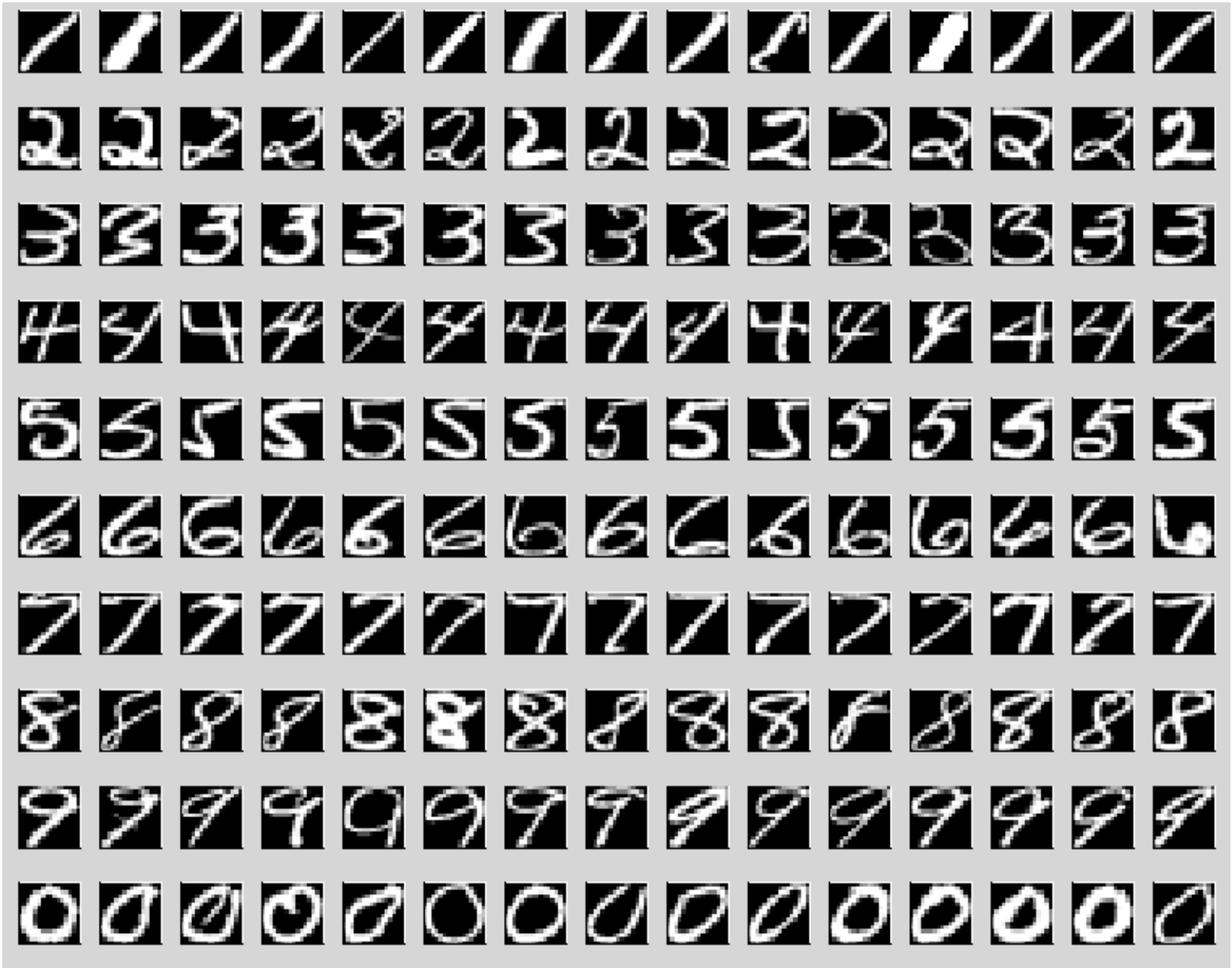
$$\hat{\mathbf{X}}_i = \sum_{j=1}^{k}(\mathbf{X}_i^T\mathbf{u}_j)\mathbf{u}_j$$

- Don't forget: The k-dimensional $\mathbf{Y}_i \;=\; \mathbf{U}_1^T\mathbf{X}_i$ is the "feature"
  - What goes into classification, or whatever
  - Use the above result to move back into the original domain

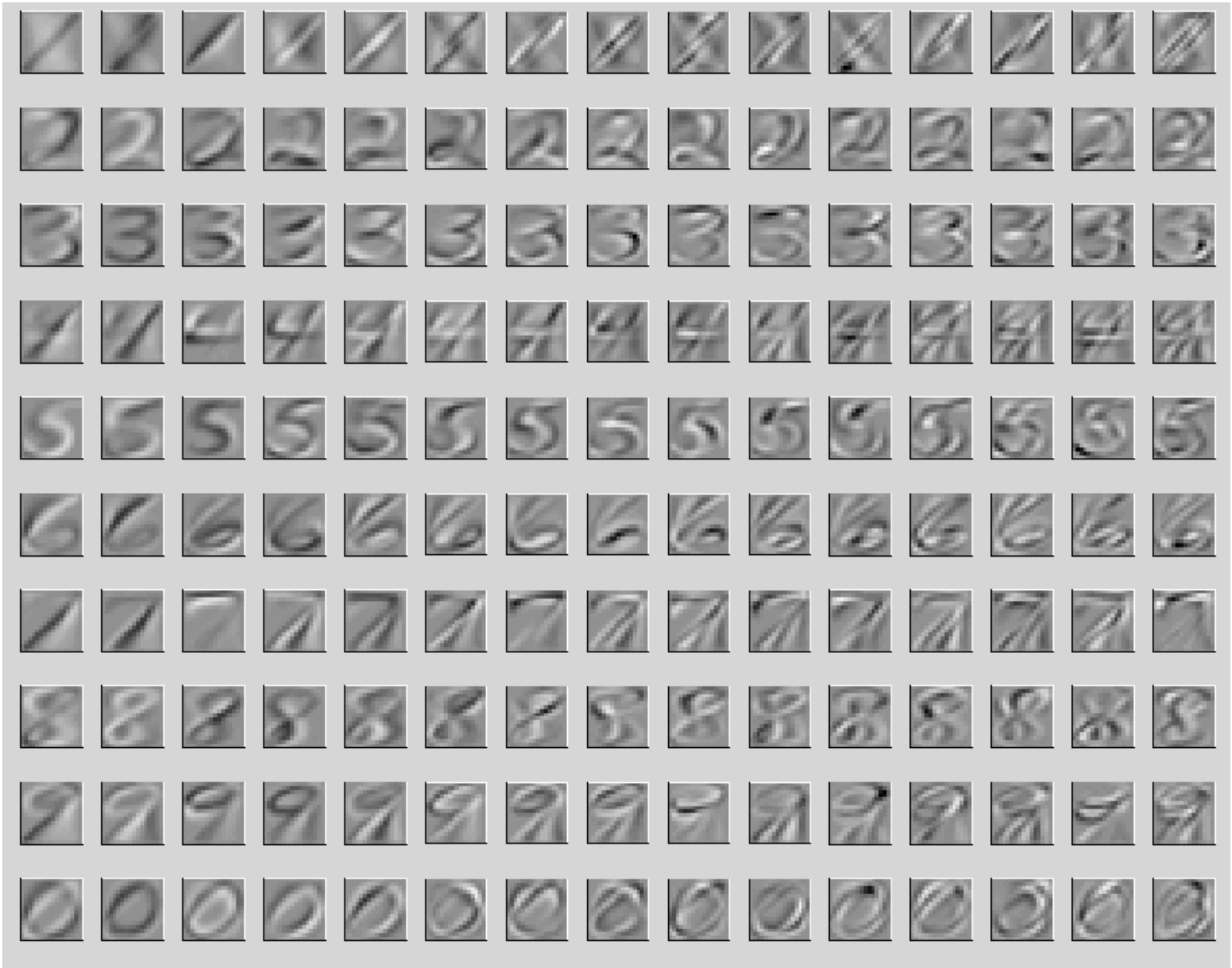
Application: Handwritten Digits

- PCA on handwritten digits
  - Length-256 data vectors (16×16 pixel grayscale images)
  - Full data has 1,100 cases on each of 10 digits
- Data reduction
  - Do we really need 256 dimensions to represent each observation?
  - How many do we need?

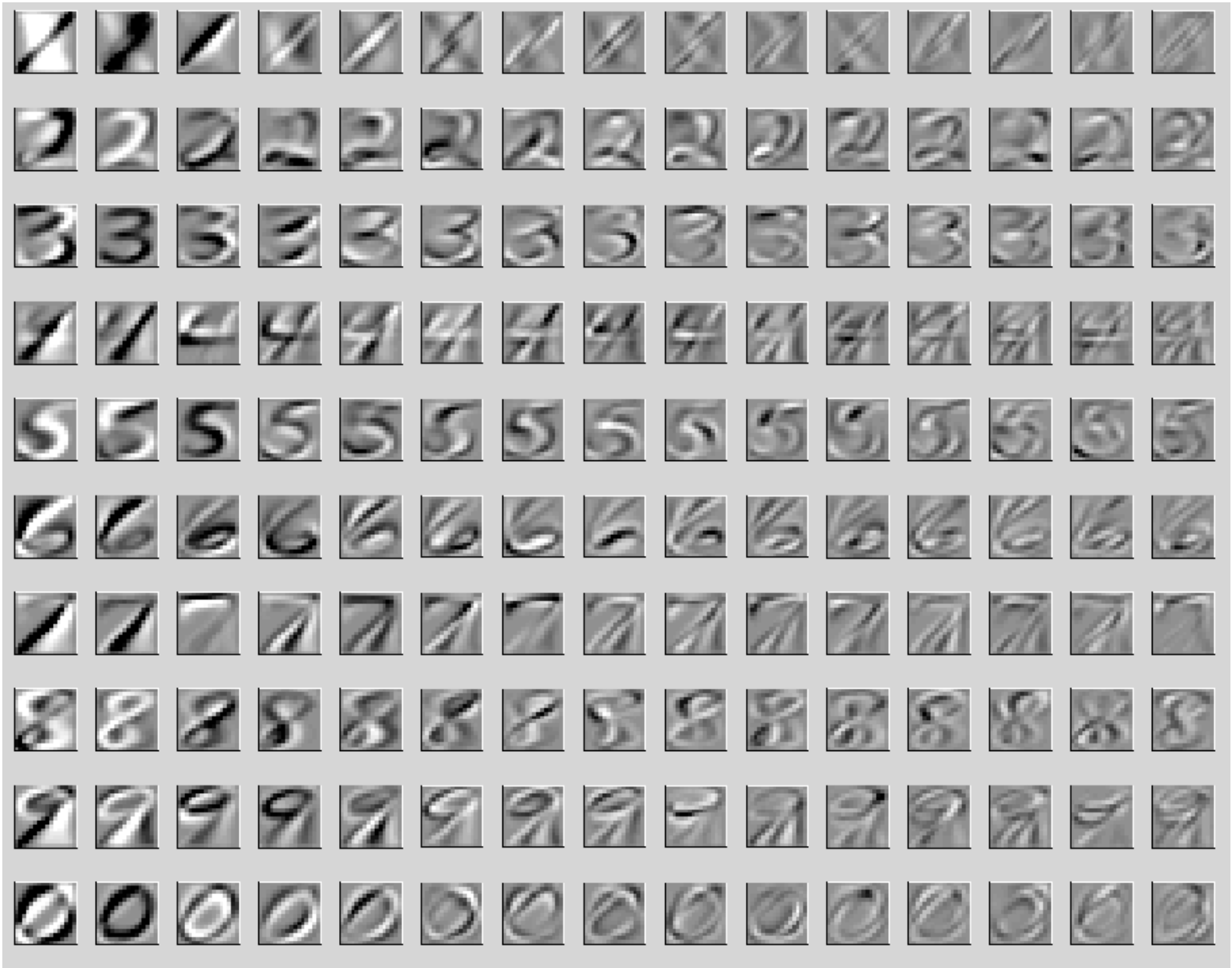
Digits: First 15 cases of 1,100



Digits: First 15 eigenvectors of 1,100



Eigenvectors scaled by $\sqrt{\lambda_j}$



Recall sample covariance of $\mathbf{U}'\mathbf{X}$ for $k=d$?

Approach

For each digit, 0, 1, 2,…

X = data, 256 × 1,100 matrix

Computed mean of 1,100 cases, Xb

Computed centered data, Xc (subtract off pixel-wise mean)

Compute S = Xc * Xc' / 1100

Compute eigenspectrum of S

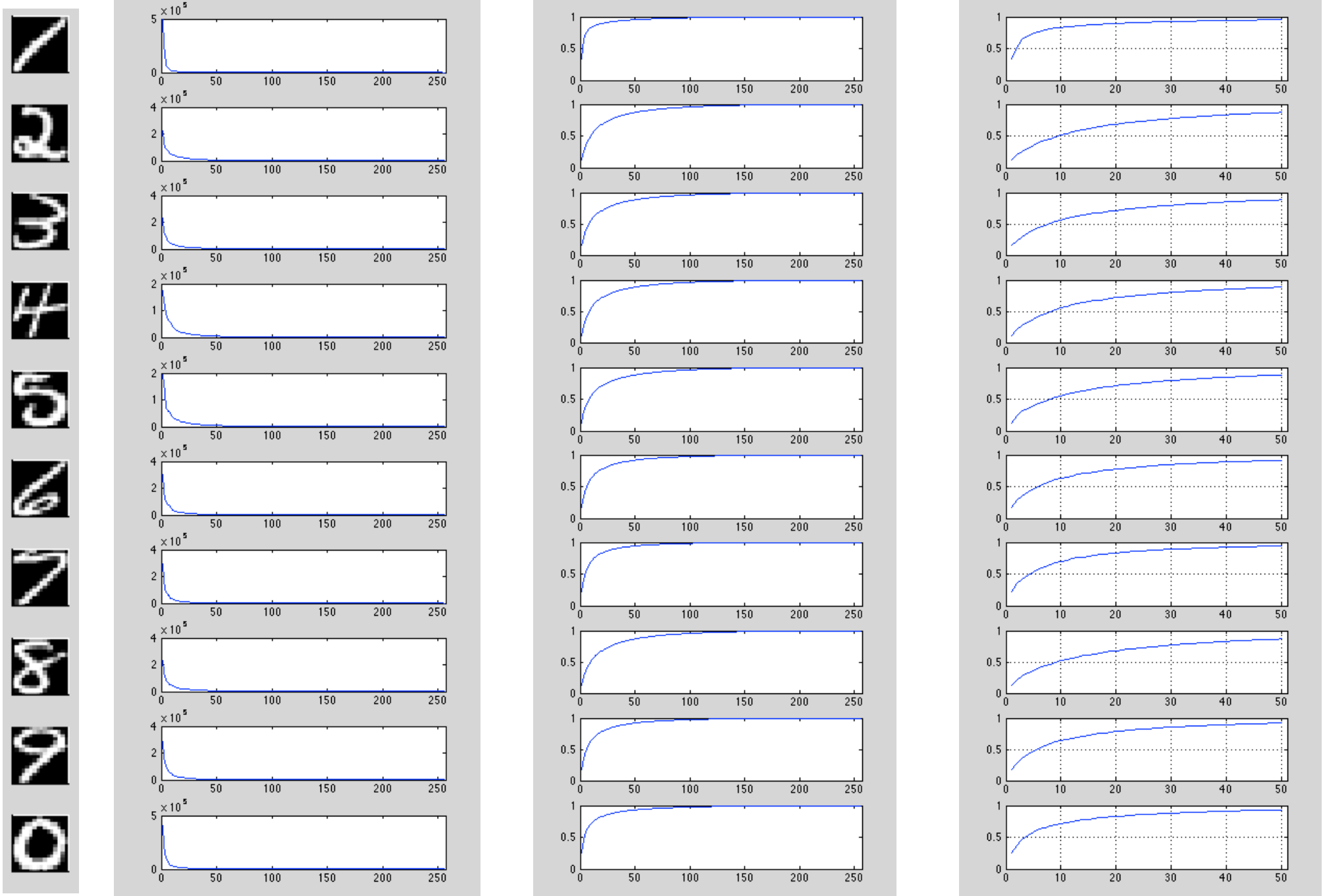
Reconstruct data for different k

Eigenspectrum

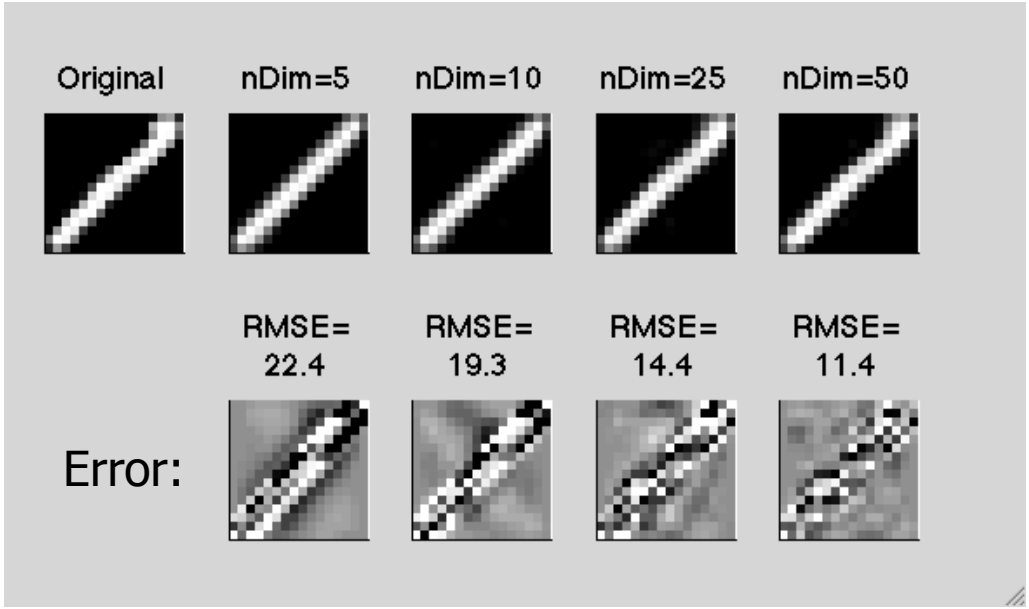
λ_j – raw values

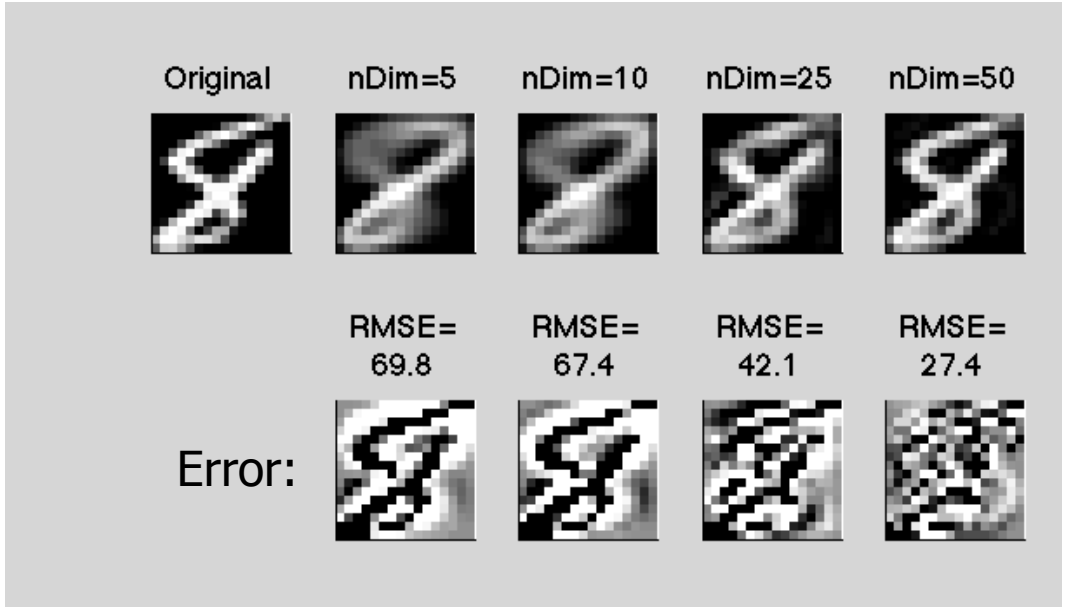
Cumulative sum, normalized to 1

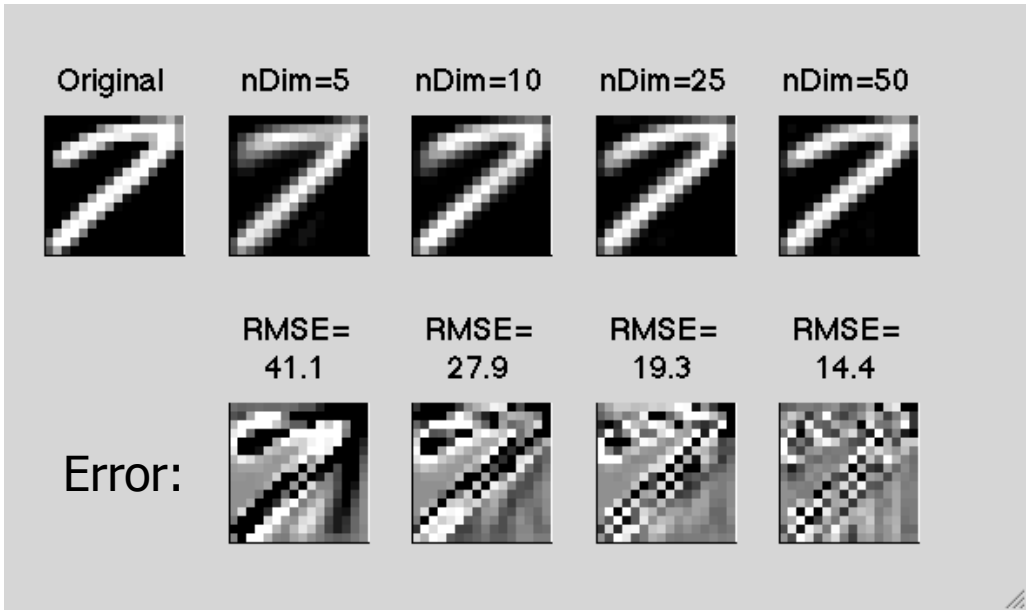
Cumulative sum, zoomed in

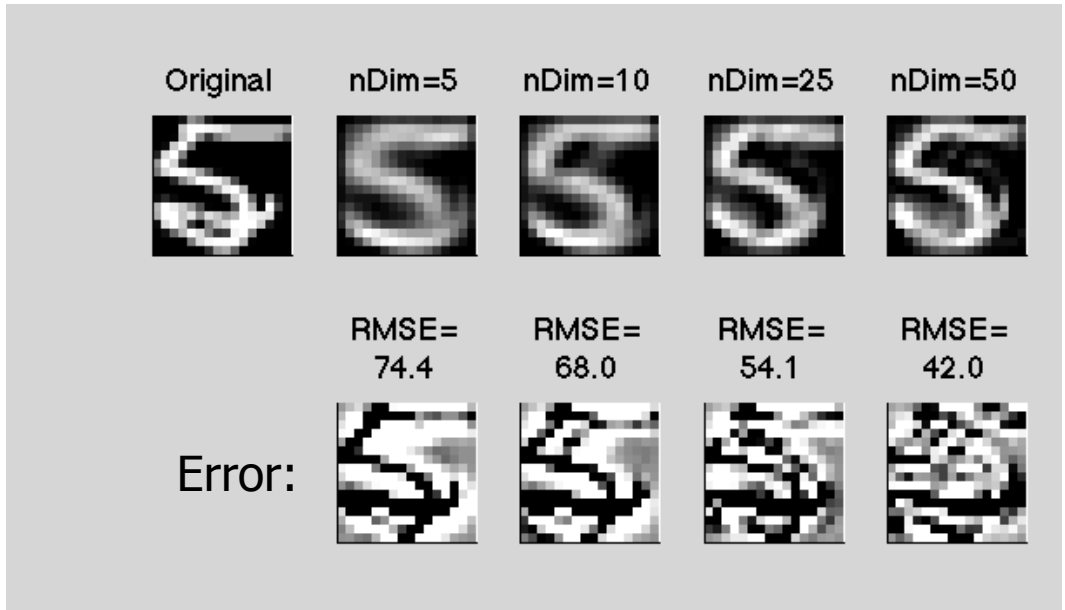


Approximations of varying k









Error images intensity range displayed: [-25, 25]

Did I cheat?

- I used n=1,100 cases on each!
  - But I only gave you 200 each in lab ☺
- Could you have done PCA?
  - If, e.g., n = 100, what is the rank of S?
  - What will the eigenspectrum look like?

SVD for PCA

- Singular Value Decomposition
  - Factorisation for arbitrary (non-square) matrices
  - For $n \times d$ matrix $\mathbf{X}$

    $$\mathbf{X} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}$$

    where

    $\mathbf{U}$ – Eigenvectors of $\mathbf{X}\mathbf{X}'$

    $\mathbf{V}$ – Eigenvectors of $\mathbf{X}'\mathbf{X}$

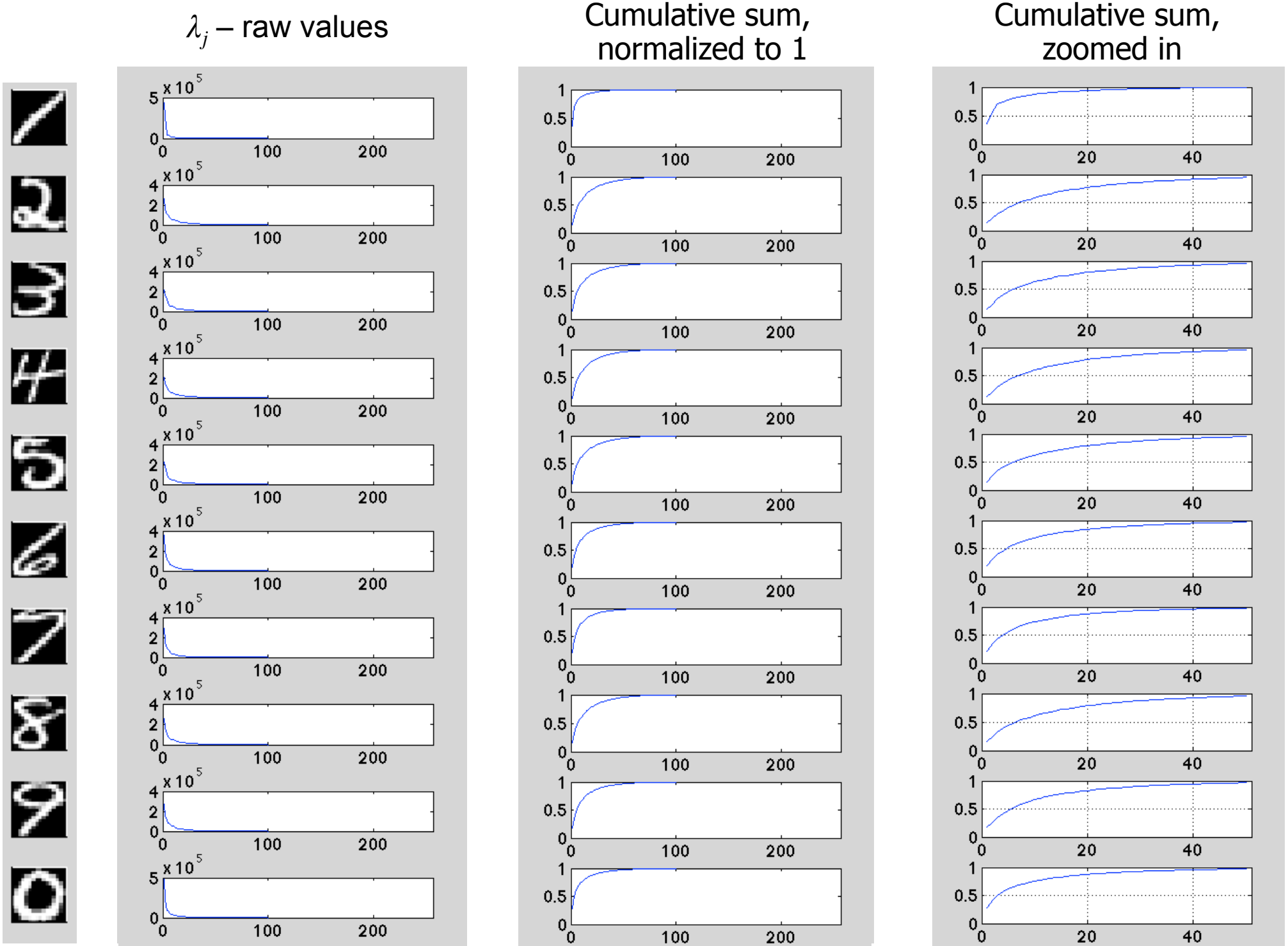
    $\mathbf{\Sigma}$ – $(\mathbf{\Sigma})_{jj} = \sqrt{\lambda_j}$ where λ_j are common eigenvalues of $\mathbf{X}\mathbf{X}'$ and $\mathbf{X}'\mathbf{X}$
- SVD can be run directly X
  - No need to make huge covariance matrix

SVD for PCA

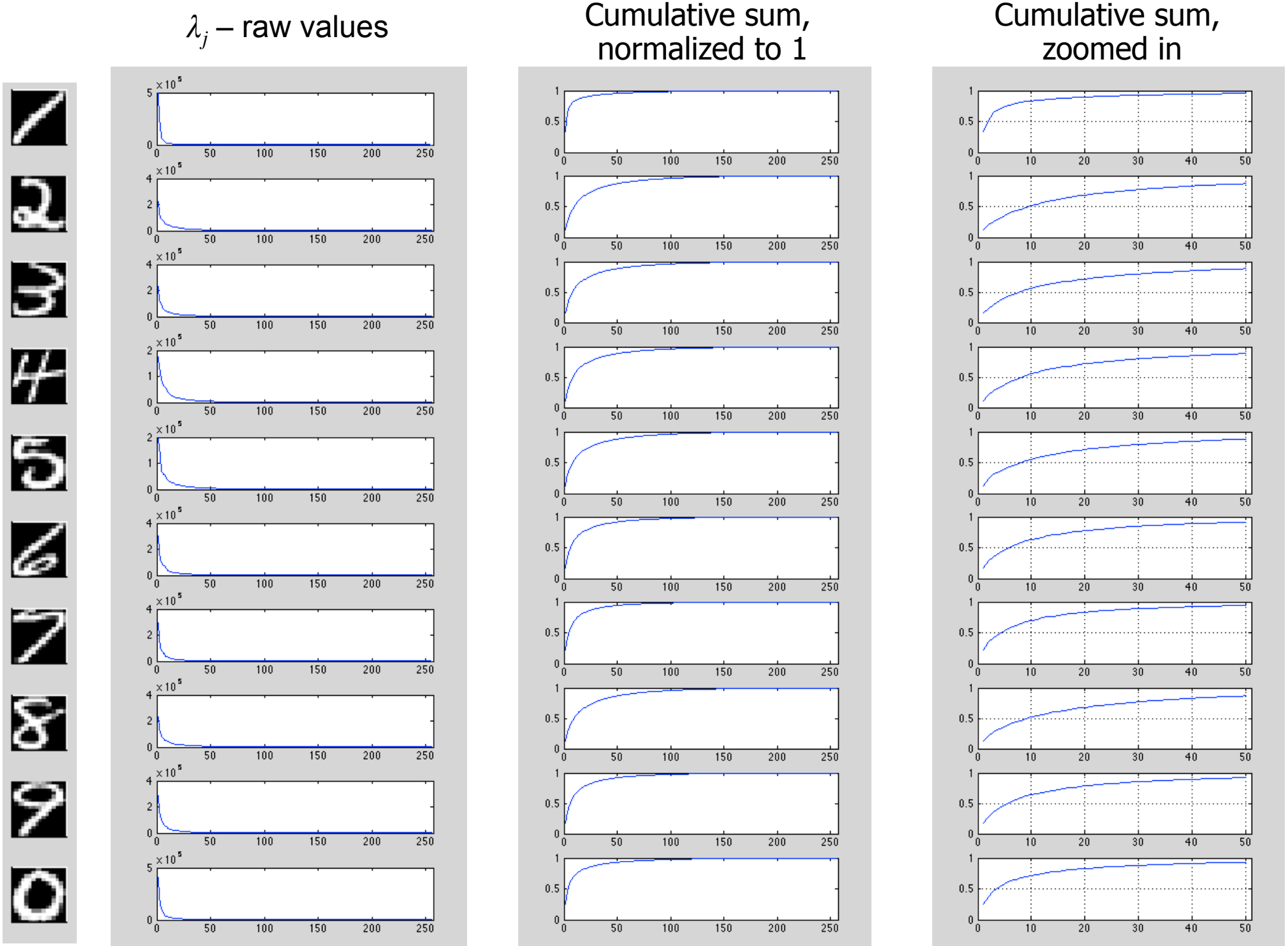
- Carefully…

- So… can either compute
  - [V, D] = eig(S);
    Col's of V eigenvectors of S
    Diagonal of D eigenvalues of S (sorted ascending)
  - [U, S] = svd(Xcenter)
    Col's of U eigenvectors of S
    diag(S).^2/n eigenvalues of S (sorted descending)

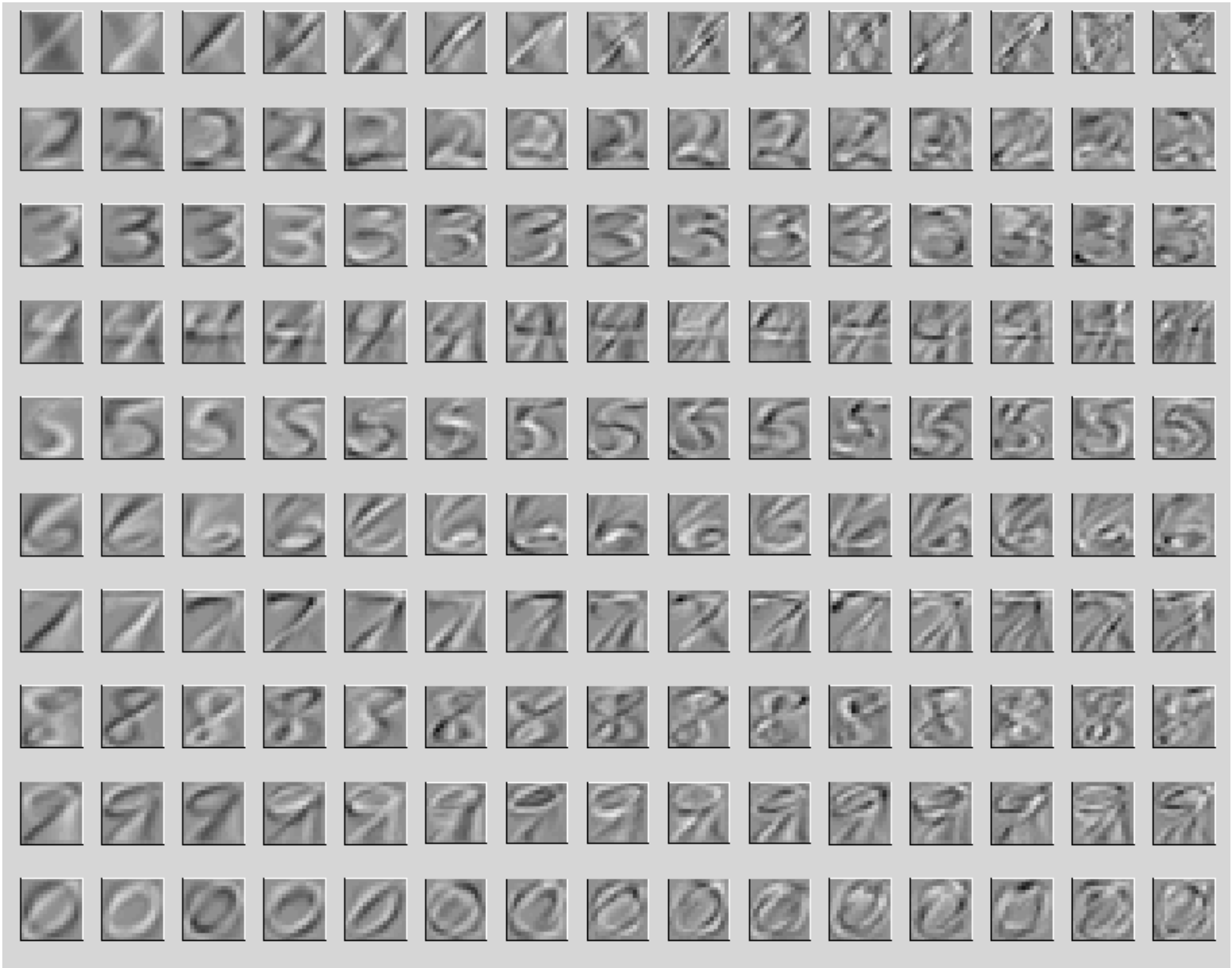
Eigenspectrum… n = 100



Eigenspectrum… n=1100

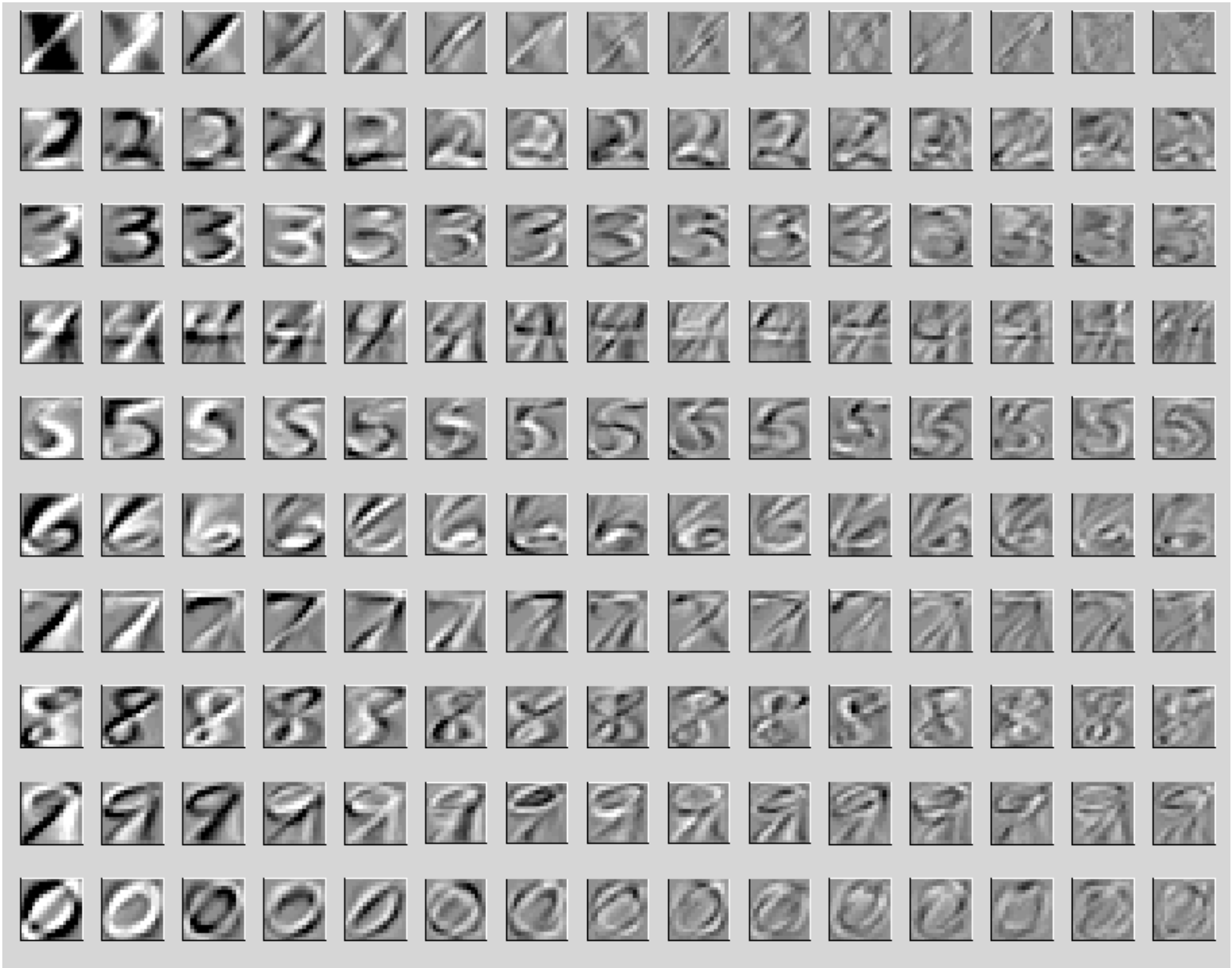


Digits: First 15 eigenvectors of 100



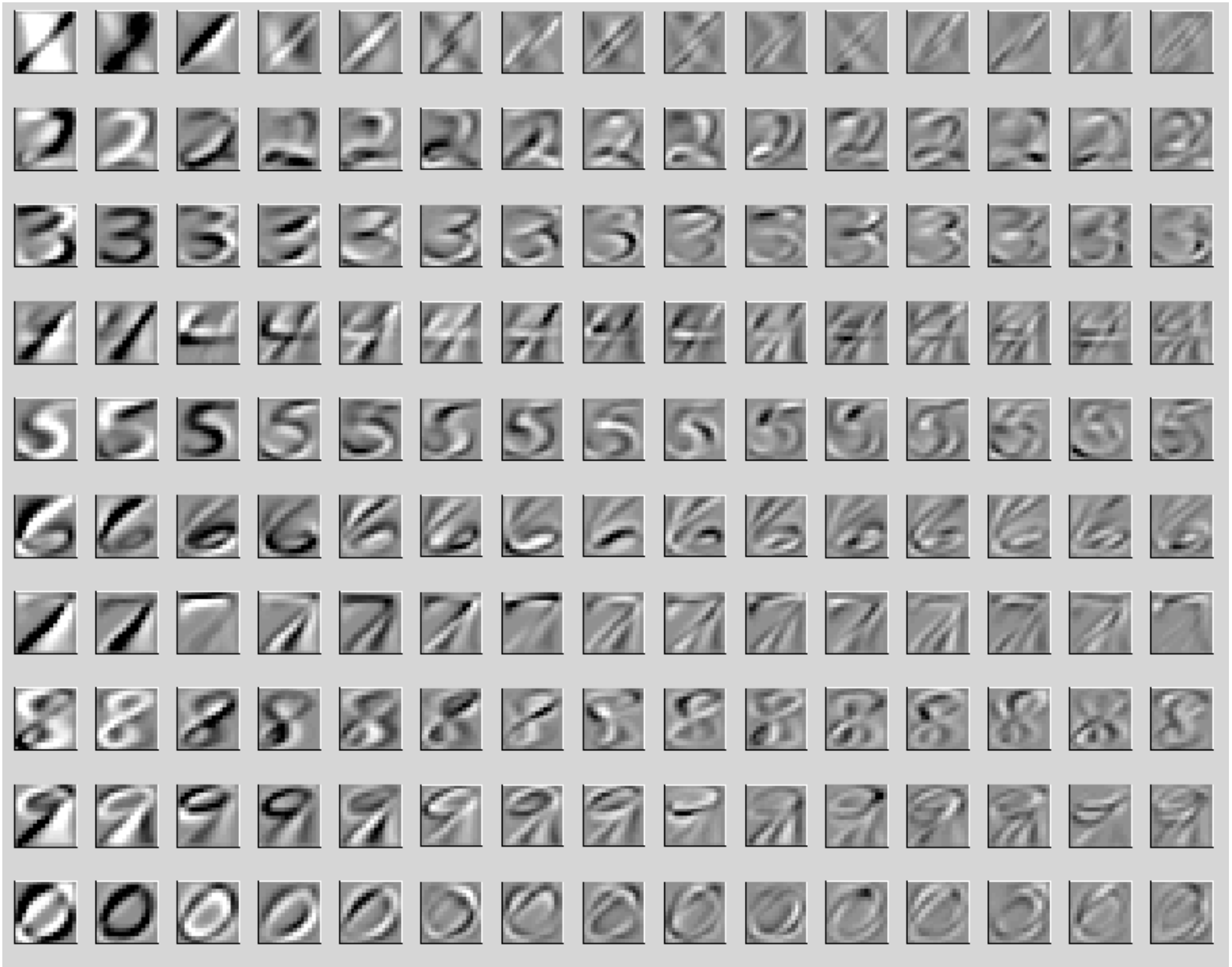
Based on n = 100

Eigenvectors scaled by $\sqrt{\lambda_j}$



Based on n = 100

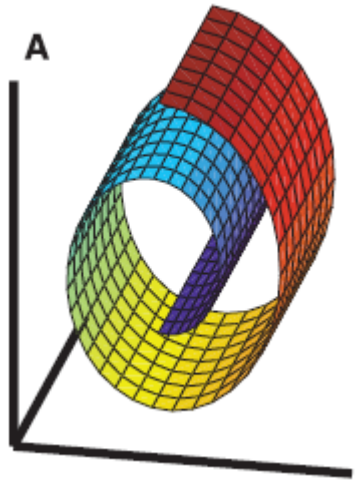
Eigenvectors scaled by $\sqrt{\lambda_j}$



Based on n = 1,100

PCA: limitations

- PCA is linear. Linear projections can only do so much. Consider:

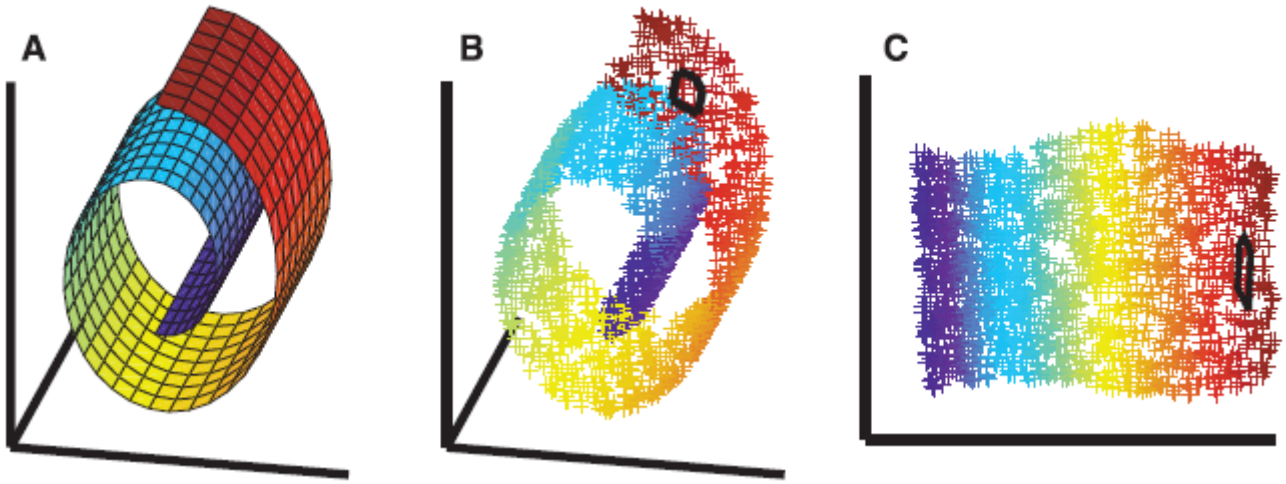


(Roweis & Saul, 2000)

- Even simpler: a circle!

Non-linear approaches

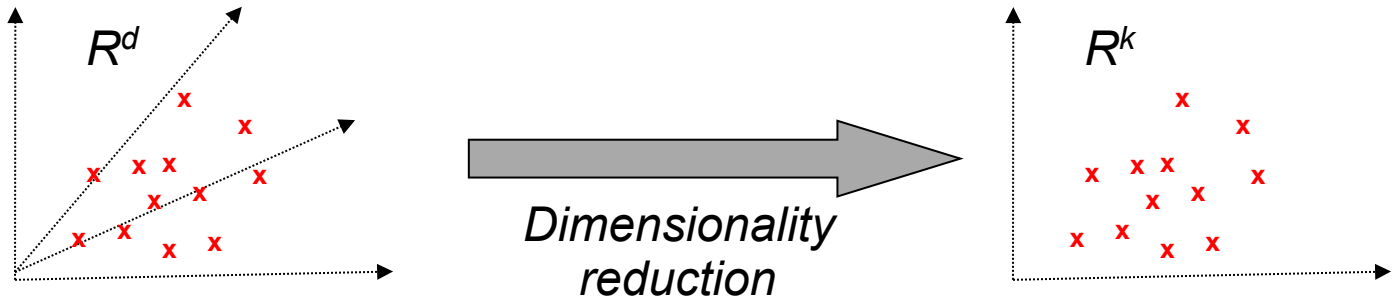
- Non-linear dimensionality reduction is hard!



Roweis & Saul, 2000)

- Locally Linear Embedding (LLE) is one alternative method
- PCA is really, really useful
  - but as always, useful to be aware of limitations, especially fundamental ones
  - Helps to diagnose what's wrong if it doesn't seem to work

Wrap up



- Can go a surprisingly long way with PCA! Basis vectors go by cool names in various fields...
  - Eigenfaces
  - Metagenes
  - Sure you can think of more... eigenpeople, anyone?
  - Key idea is that columns of $\mathbf{U_1}$ are composite dimensions which capture a lot of information and can in that sense be thought of as "meta"-dimensions