10-605/805 – ML for Large Datasets Lecture 3: Principal Component Analysis

Henry Chai

9/6/22

Front Matter

- HW1 released 8/30, due 9/14 at 11:59 PM
  - **For HW1 only, the programming part is optional (but strongly encouraged)**
  - The written part is nominally about PCA but can be solved using pre-requisite knowledge (linear algebra)
- Recitations on Friday, 11:50 – 1:10 (**different from lecture**) in GHC 4401 (**same as lecture**)
  - Recitation 2 on 9/9: Review of linear algebra

Data Pre-processing

- ETL (extract-transfer-load)
- Cleaning data
  - Missing features/labels
  - Duplicated observations
  - Formatting errors
- Understanding data
  - Summarization
  - Exploration
  - Visualization

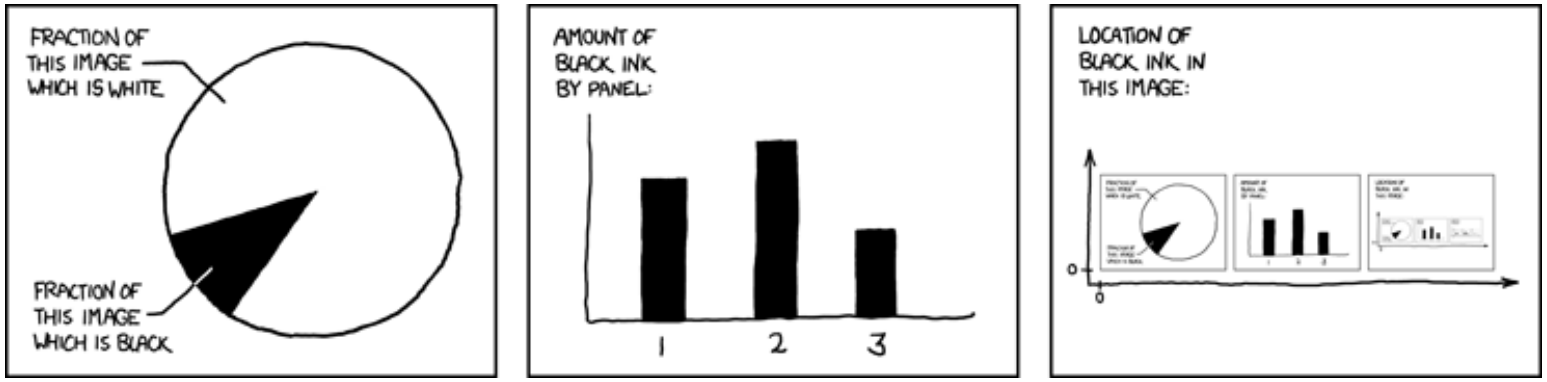
Data Pre-processing

- ETL (extract-transfer-load)
- Cleaning data
  - Missing features/labels
  - Duplicated observations
  - Formatting errors
- **Understanding data**
  - **Summarization**
  - **Exploration**
  - **Visualization**

Given some (labelled) dataset, what questions can you ask to better understand the data?

- what are the units?
- what are the features?
- and what type are they all?
- how many observations are we working w/?
- Is it complete and collected uniformly/systematically?
- are features related to one another?
- what are actually doing?

Data Visualization

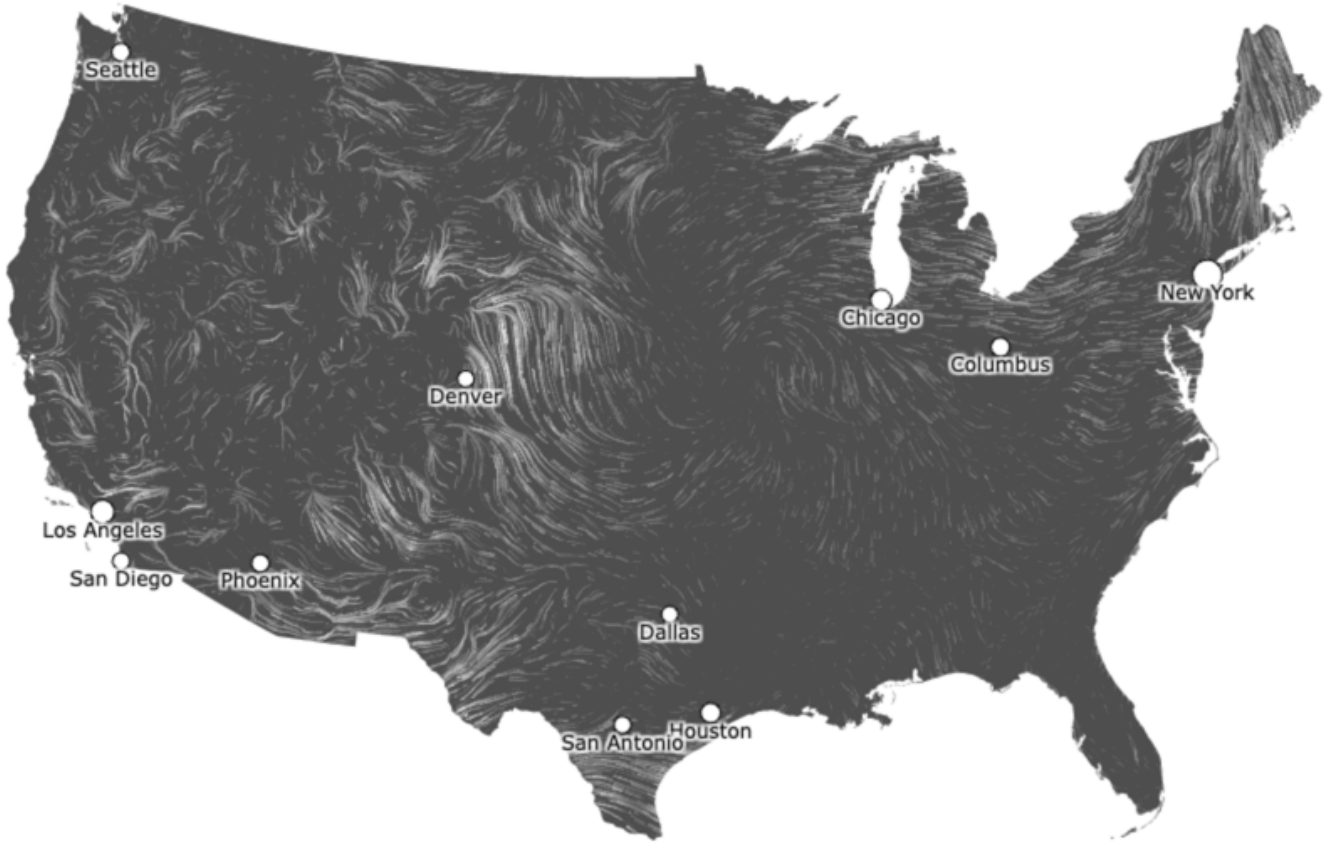


- Visualizations can be used to
  - Provide insight about trends/groups/relationships
  - Reveal systematic errors
  - Aid in model selection
  - Evaluate training (e.g., measure convergence)
  - Interpret/explain predictions

Source: https://xkcd.com/688/

Data Visualization: Examples

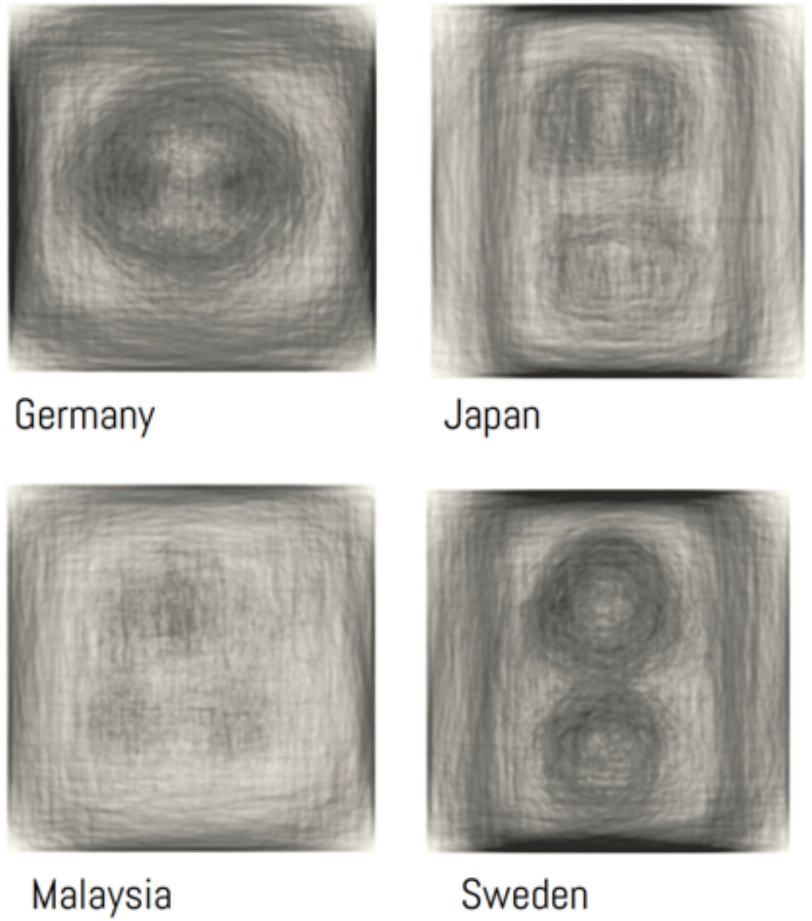
- Understanding scale



Source: http://hint.fm/wind/

Data Visualization: Examples

- Understanding clusters

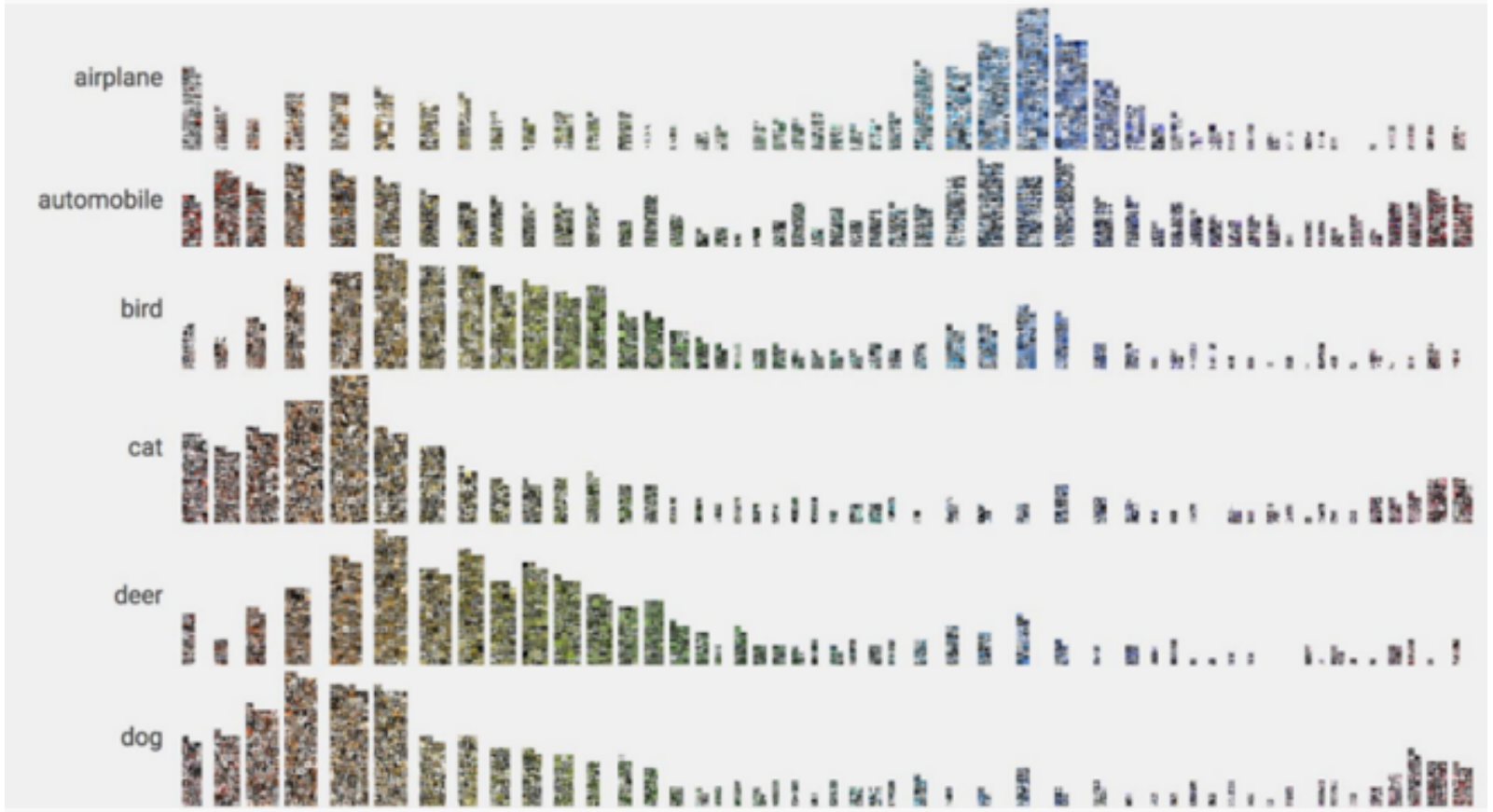


electrical outlets

Source: https://quickdraw.withgoogle.com/

Data Visualization: Examples

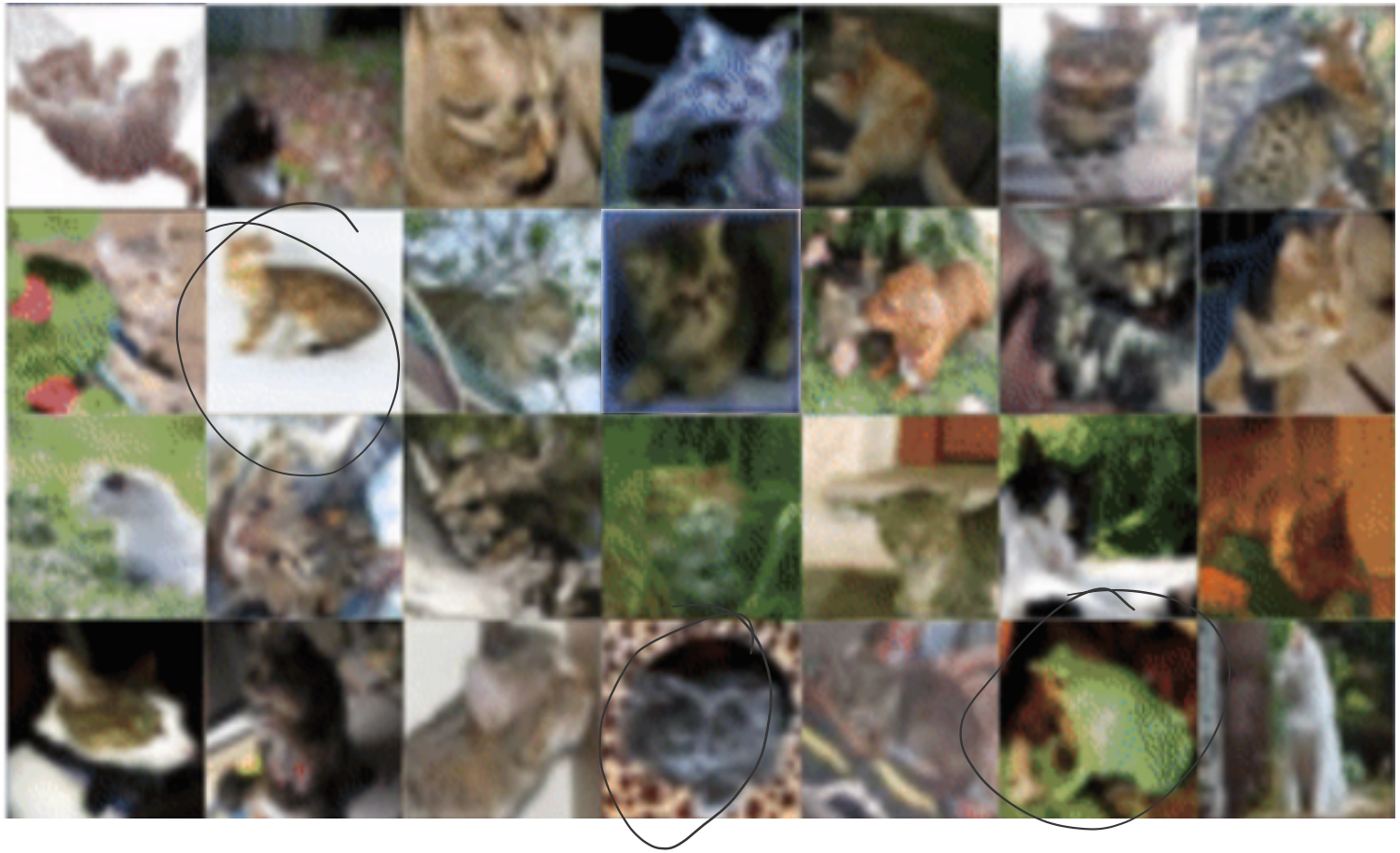
- Understanding variation



Source: https://pair-code.github.io/facets/

Data Visualization: Examples

- Understanding mistakes



Source: https://pair-code.github.io/facets/

Common Data Visualizations

- Summary statistics
- Box plots
- Histograms
- Scatter plots

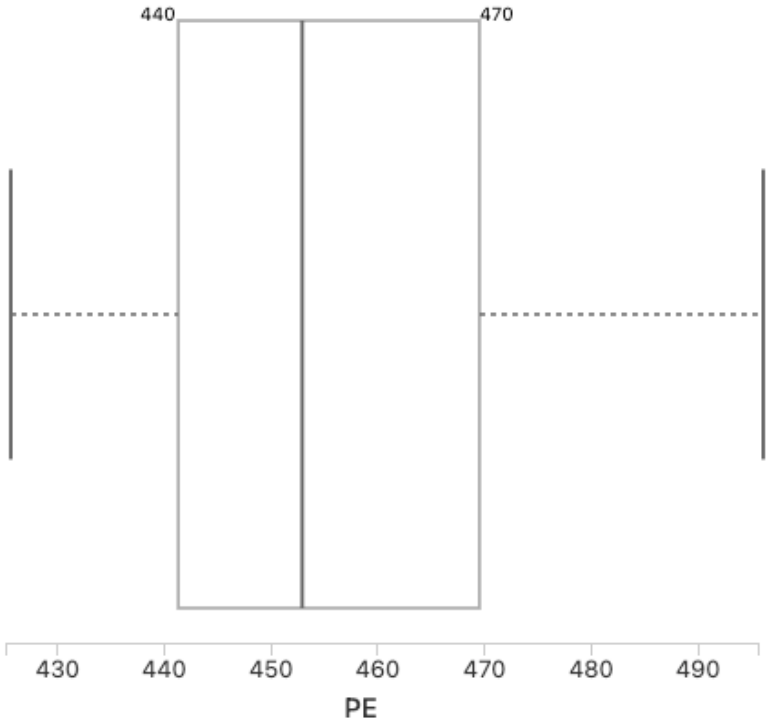
Common Data Visualizations

- Summary statistics
- Box plots
- Histograms
- Scatter plots

	AT	PE
Mean	19.65	454.37
Stddev	7.45	17.07
Min	1.81	420.26
Max	37.11	495.76

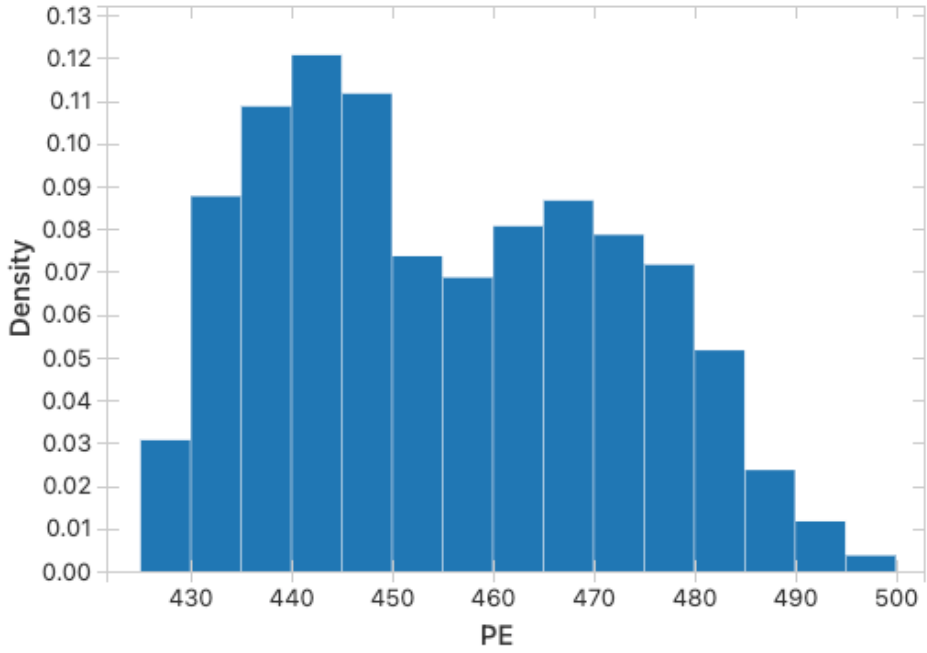
Common Data Visualizations

- Summary statistics
- Box plots
- Histograms
- Scatter plots



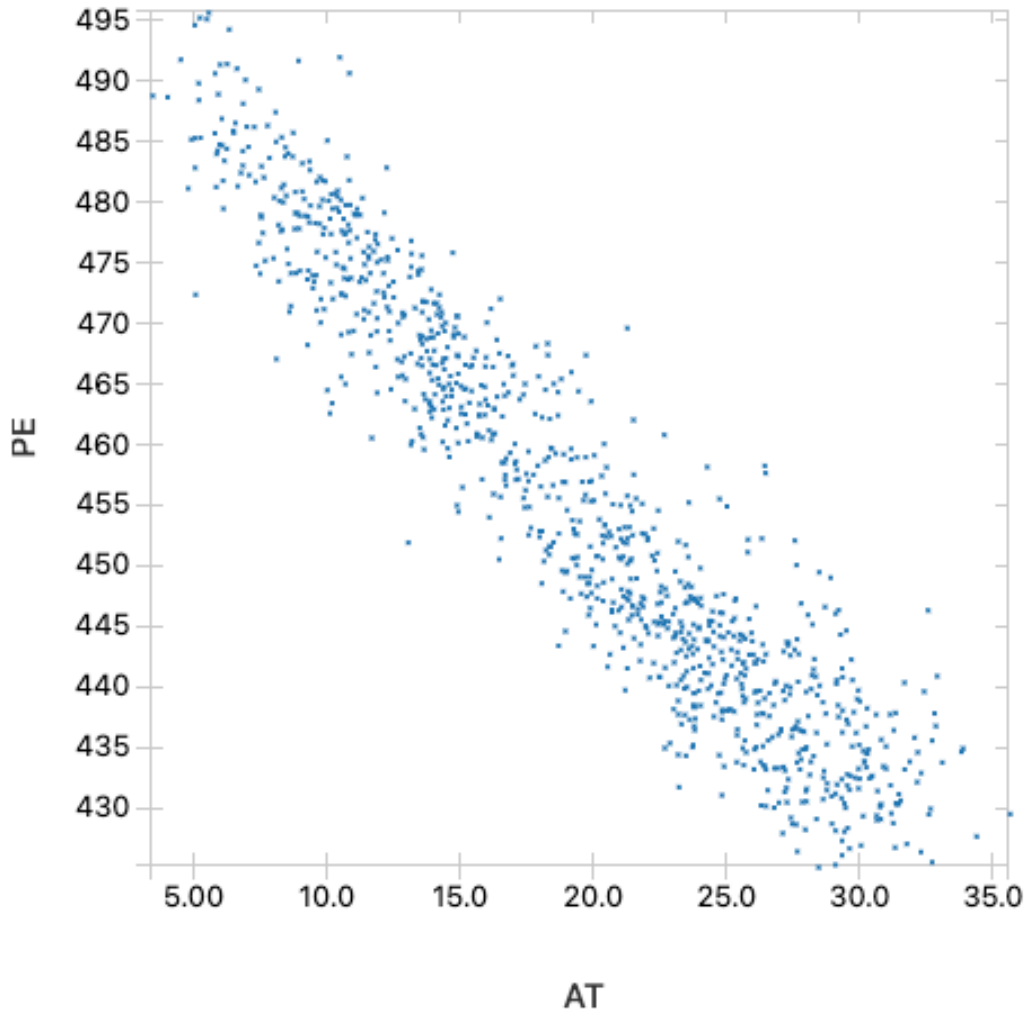
Common Data Visualizations

- Summary statistics
- Box plots
- Histograms
- Scatter plots



Common Data Visualizations

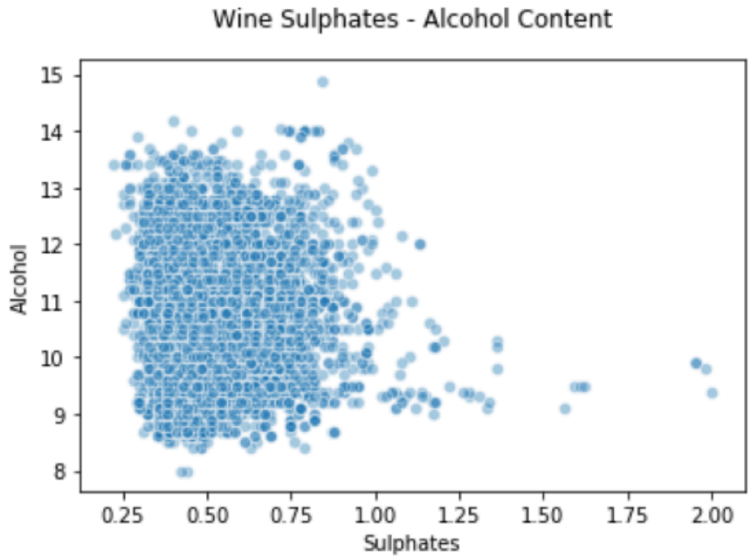
- Summary statistics
- Box plots
- Histograms
- Scatter plots

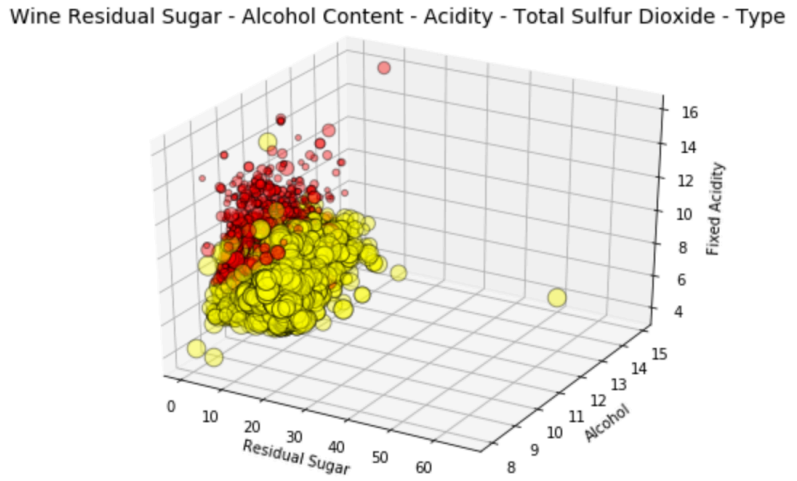


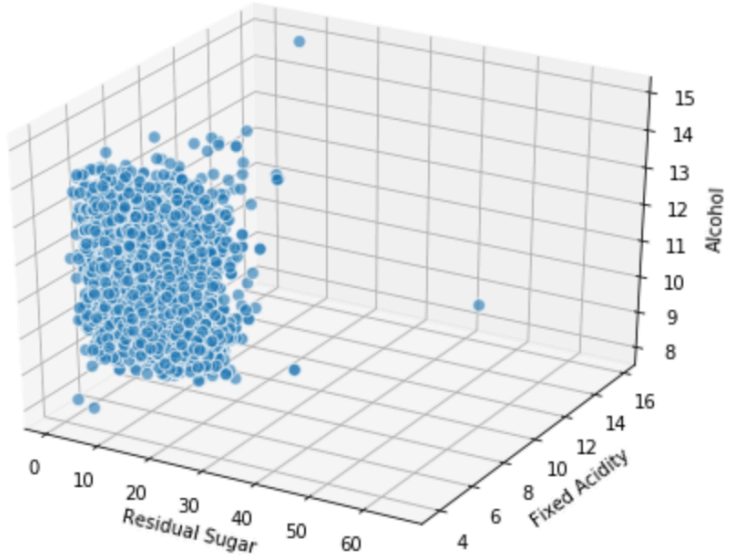
Big Data Visualizations

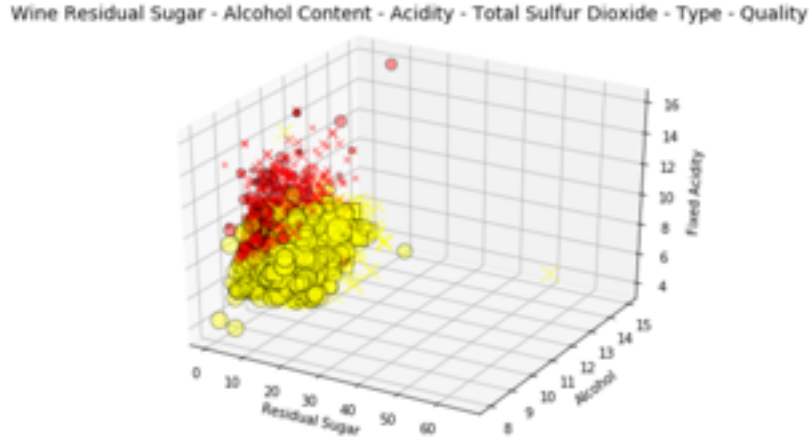
- Large n
  - Computationally expensive to render
  - Dense/complex
  - Address via subsampling or parallelization
- Large k
  - Difficult to represent more than a few dimensions

Big Data Visualizations



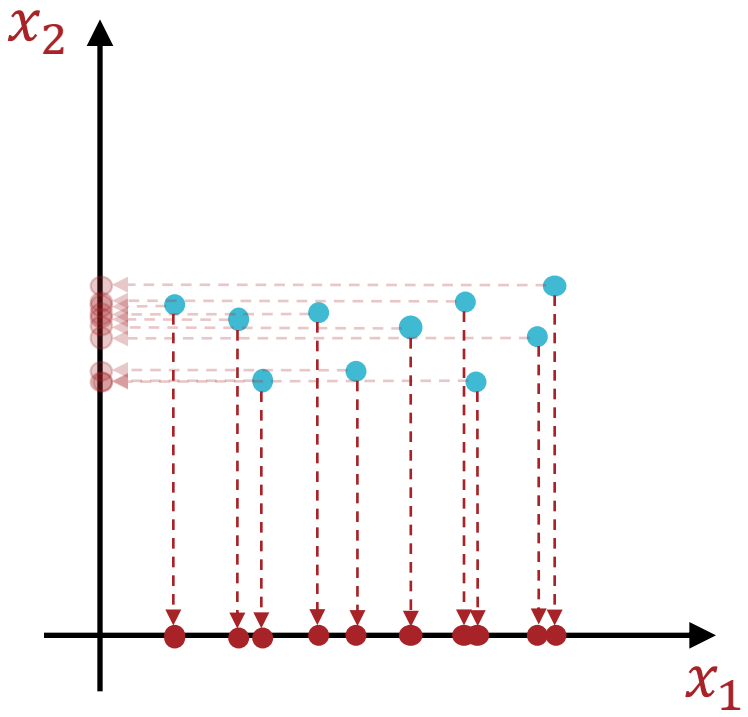


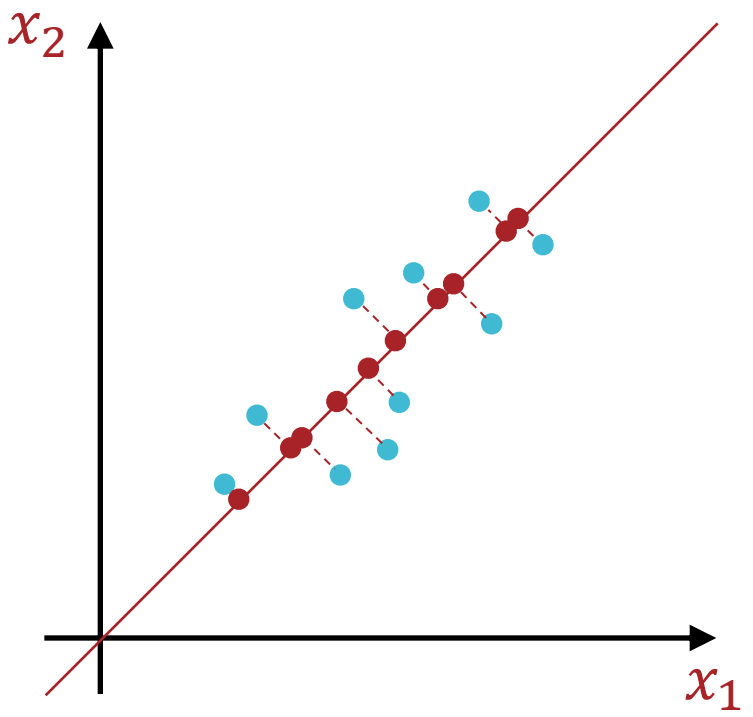




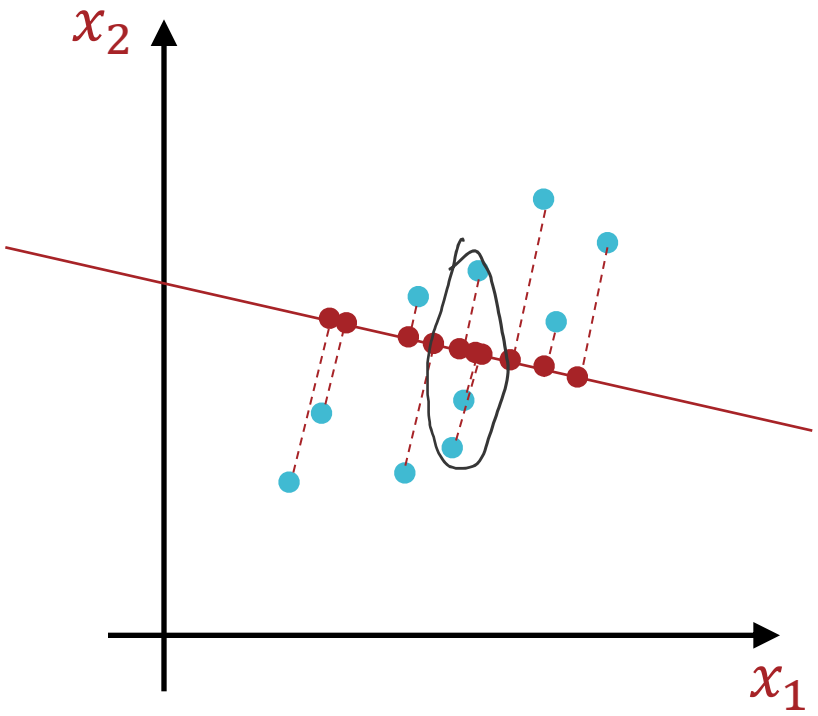
Big Data Visualizations

- Large n
  - Computationally expensive to render
  - Dense/complex
  - Address via subsampling or parallelization
- Large k
  - Difficult to represent more than a few dimensions
  - Address via dimensionality reduction = learning a latent (typically lower-dimensional) representation

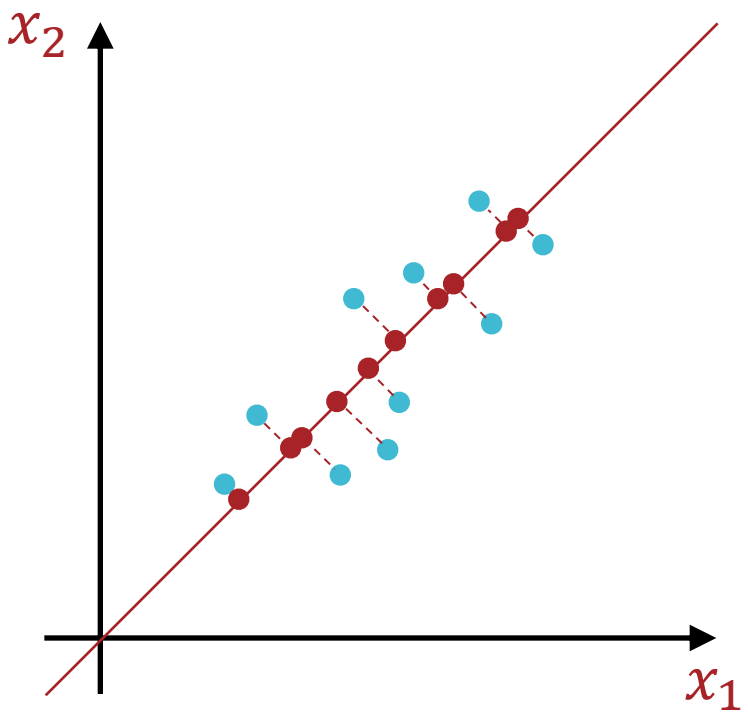




Feature Elimination

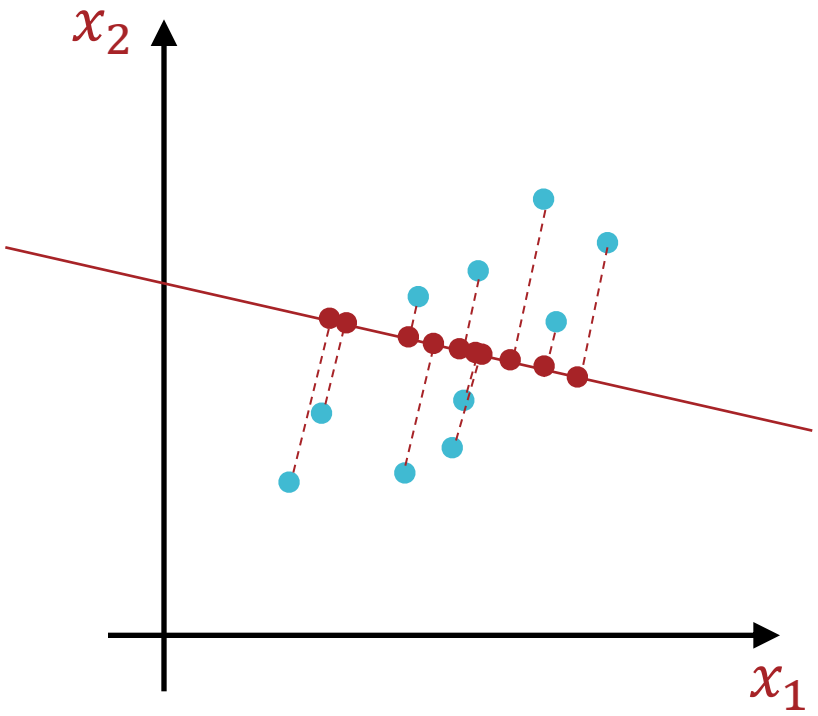


Option A

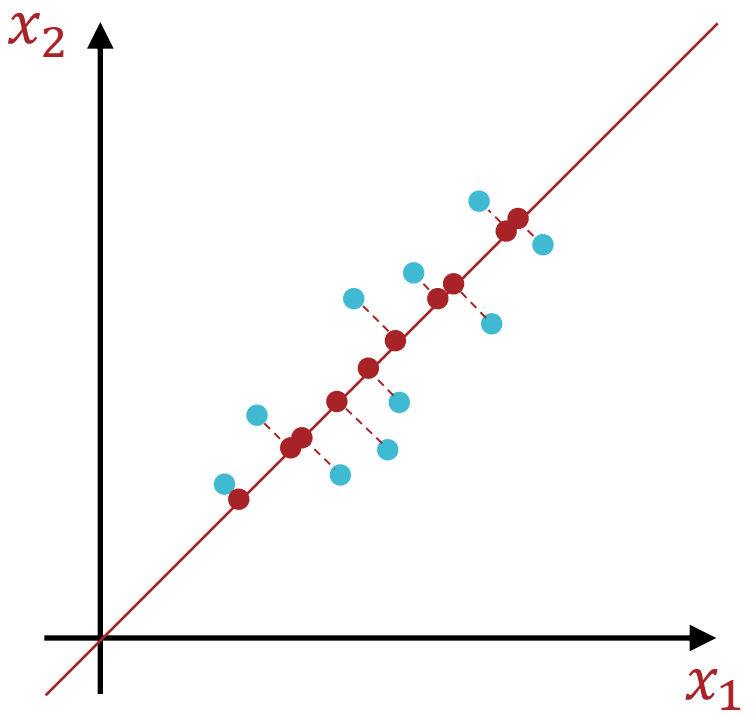


Option B

Dimensionality Reduction

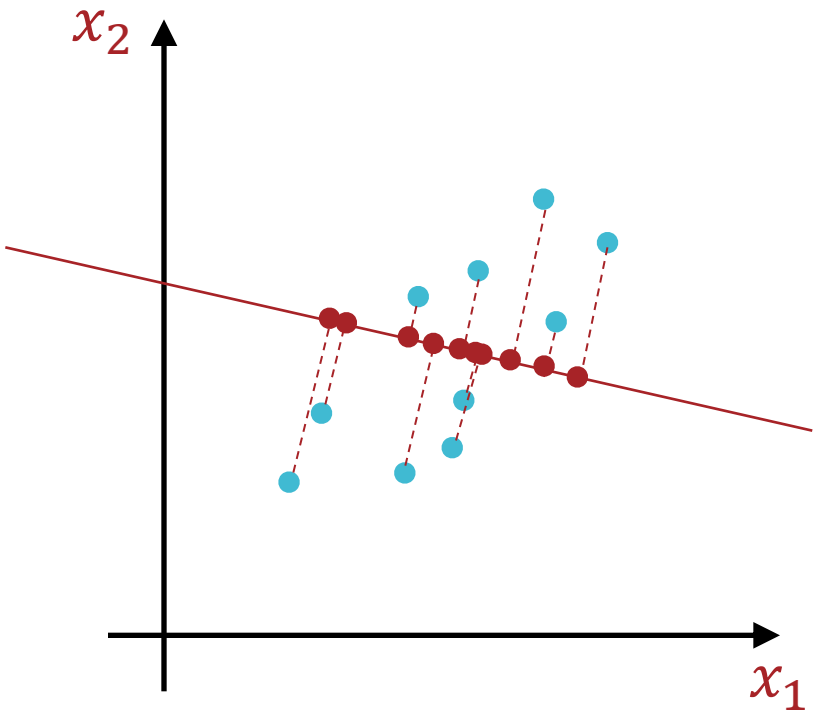


Option A

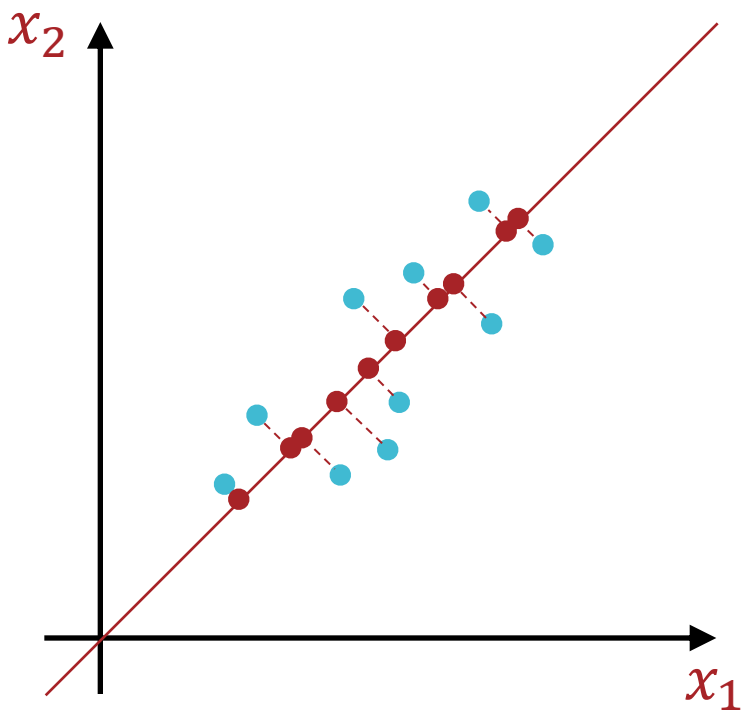


Option B

Which projection do you prefer?

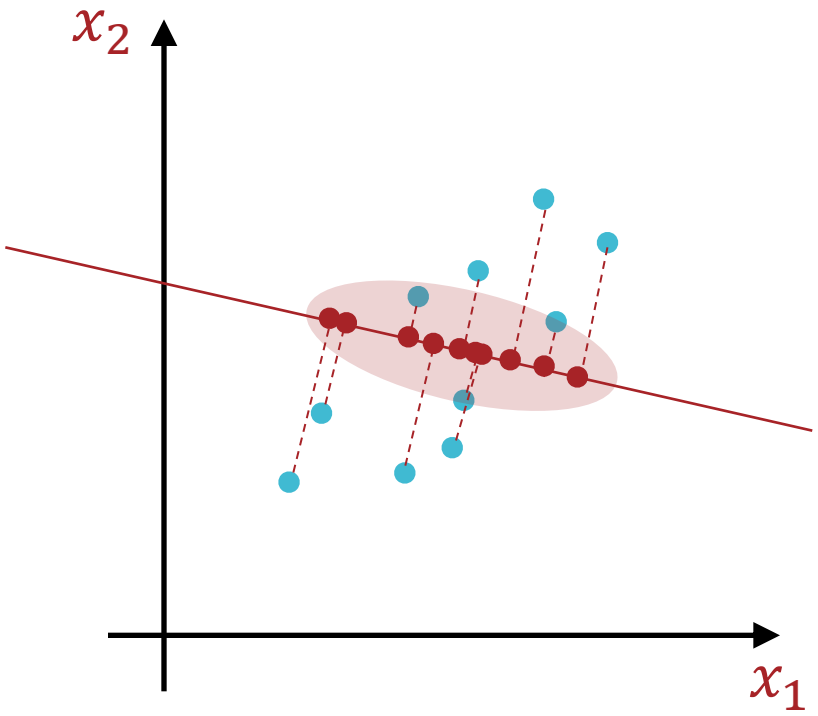


Option A

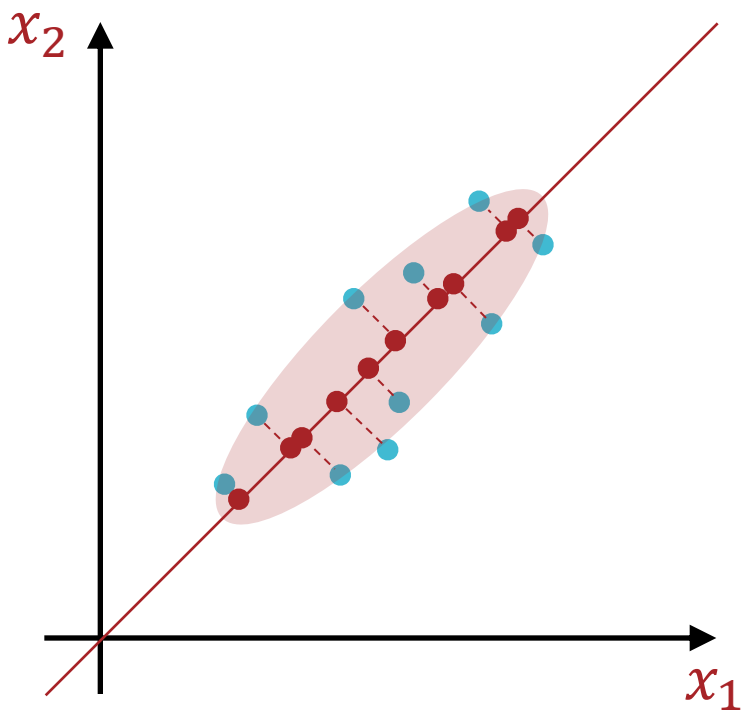


Option B

Goal: minimize the reconstruction error



Option A



Option B

Goal: maximize the variance of the projections

Centering the Data

- To be consistent, we will constrain principal components to be *orthonormal vectors* (orthogonal unit vectors) that begin at the origin
- Preprocess data to be centered around the origin:

1. $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}$
2. $\tilde{\boldsymbol{x}}^{(i)} = \boldsymbol{x}^{(i)} - \boldsymbol{\mu}\ \forall\ i$
3. $X = \underbrace{\begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)^T} \\ \tilde{\boldsymbol{x}}^{(2)^T} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(n)^T} \end{bmatrix}}_{} \in \mathbb{R}^{n \times k}$

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(i)}$ onto a vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(i)} = \left(\frac{\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(i)}}{\|\boldsymbol{v}\|_2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Length of projection

Direction of projection

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(i)}$ onto a unit vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(i)} = \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(i)}\right)\boldsymbol{v}$$

$$\hat{v} = \underset{v:\|v\|_2=1}{\operatorname{argmin}} \left(\text{the reconstruction error}\right)$$

$$= \underset{v:\|v\|_2=1}{\operatorname{argmin}} \sum_{i=1}^{n} \|\tilde{x}^{(i)} - z^{(i)}\|_2^2$$

$$= \underset{v:\|v\|_2=1}{\operatorname{argmin}} \sum_{i=1}^{n} \|\tilde{x}^{(i)} - (v^T\tilde{x}^{(i)})v\|_2^2$$

$$\|\tilde{x}^{(i)} - (v^T\tilde{x}^{(i)})v\|_2^2 = \left(\tilde{x}^{(i)T}\tilde{x}^{(i)} - 2v^T\tilde{x}^{(i)}\left(v^T\tilde{x}^{(i)}\right) + v^T\tilde{x}^{(i)}\left(v^T\tilde{x}^{(i)}\right)\underbrace{v^T v}_{\|v\|_2^2=1}\right)$$

$$= \tilde{x}^{(i)T}\tilde{x}^{(i)} - v^T\tilde{x}^{(i)}\left(v^T\tilde{x}^{(i)}\right)$$

Minimizing the Reconstruction Error

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{i=1}^{n} \left\|\tilde{\boldsymbol{x}}^{(i)}\right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(i)}\right)^2$$

$$\hat{v} = \underset{v:\|v\|_2=1}{\operatorname{argmin}} \sum_{i=1}^{n} -\left(v^T \tilde{x}^{(i)}\right)^2 = \underset{v:\|v\|_2=1}{\operatorname{argmax}} \sum_{i=1}^{n} \left(v^T \tilde{x}^{(i)}\right)^2$$

$$E\left[v^T \tilde{x}^{(i)}\right] = 0$$

$$= \underset{v:\|v\|_2=1}{\operatorname{argmax}} \sum_{i=1}^{n} \left(v^T \tilde{x}^{(i)} \tilde{x}^{(i)T} v\right) = \underset{v:\|v\|_2=1}{\operatorname{argmax}} v^T \left(\sum_{i=1}^{n} \tilde{x}^{(i)} \tilde{x}^{(i)T}\right) v$$

$$= \underset{v:\|v\|_2=1}{\operatorname{argmax}} \; v^T X^T X v$$

$$= \underset{v:\|v\|_2=1}{\operatorname{argmax}} \; v^T C_x v$$

C_x → covariance matrix

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T C_X \boldsymbol{v}$$

$$L(v, \lambda) = v^T C_x v - \lambda\left(\|v\|_2^2 - 1\right)$$

$$= v^T C_x v - \lambda\left(v^T v - 1\right)$$

$$\frac{\partial L}{\partial v} = C_x v - \lambda v$$

$$\Rightarrow C_x \hat{v} - \lambda \hat{v} = 0 \Rightarrow C_x \hat{v} = \lambda \hat{v}$$

λ is an eigenvalue of C_x and its corresponding eigenvector $\hat{v}$

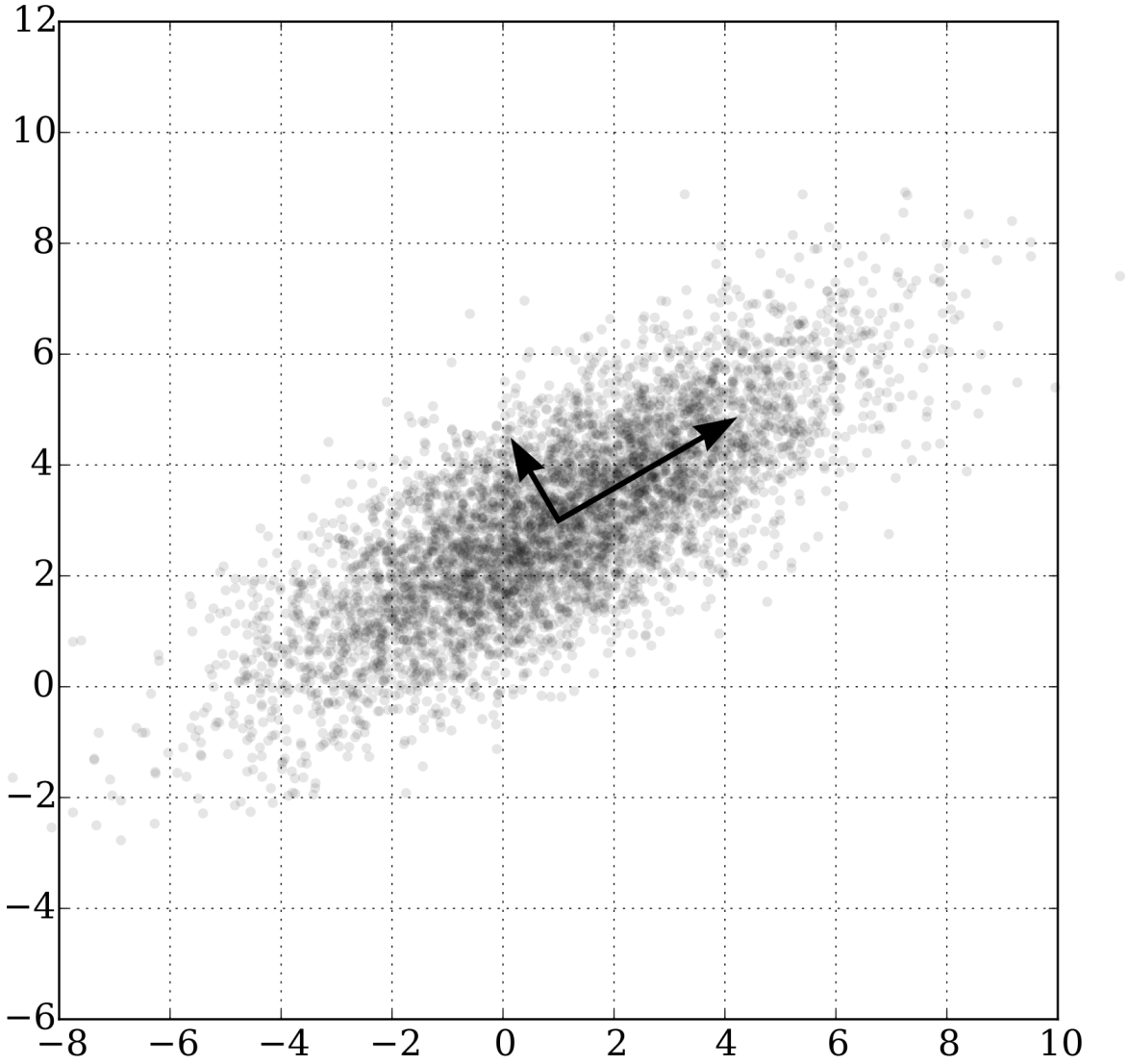
Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T C_X \boldsymbol{v}$$

$$C_X\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}} \rightarrow \hat{\boldsymbol{v}}^T C_X\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}} = \lambda$$

- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_2
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthonormal!
- Etc …
- λ_i is a measure of how much variance falls along $\hat{\boldsymbol{v}}_i$

Principal Components: Example



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

HW1 Aside: Sparse PCA

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1 \text{ and } \|\boldsymbol{v}\|_0 \leq s}{\operatorname{argmax}} \boldsymbol{v}^T C_X \boldsymbol{v}$$

- L0-norm constraint: $\boldsymbol{v}$ can contain at most s non-zero elements
- Reduces to standard PCA if $s = k$
- Sparse principal components may be easier to interpret and can also reduce data pre-processing needs

PCA Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. Center the data
   A. Optionally, normalize the data by features so that all features are of the same scale
2. Compute the covariance matrix $C_X = X^T X$
3. Collect the top r eigenvectors (corresponding to the r largest eigenvalues), $P \in \mathbb{R}^{k \times r}$
4. Project the data into the space defined by P, $Z = XP$

- Output: Z, the latent representation ("PCA scores")

How many PCs should we use?

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. Center the data
   A. Optionally, normalize the data by features so that all features are of the same scale
2. Compute the covariance matrix $C_X = X^T X$
3. Collect the top r eigenvectors (corresponding to the r largest eigenvalues), $P \in \mathbb{R}^{k \times r}$
4. Project the data into the space defined by P, $Z = XP$

- Output: Z, the latent representation ("PCA scores")

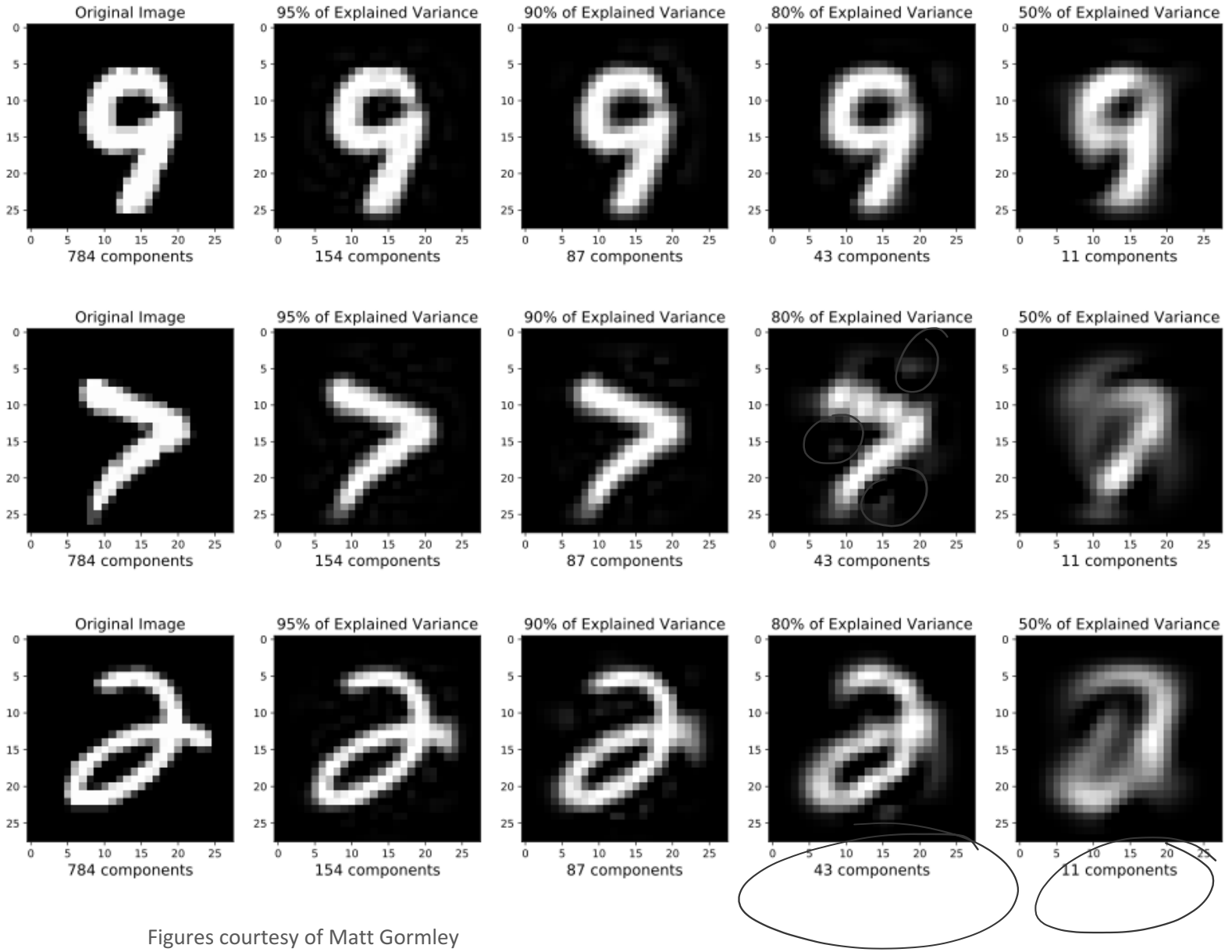
Choosing the number of PCs

- Define a percentage of explained variance for the i^{th} PC:

$$\lambda_i \Big/ \sum \lambda_j$$

- Select all PCs above some threshold of explained variance, e.g., 5%
- Keep selecting PCs until the total explained variance exceeds some threshold, e.g., 90%
- Evaluate on some downstream metric

PCA Example: MNIST Digits



Figures courtesy of Matt Gormley

PCA Example: MNIST Digits

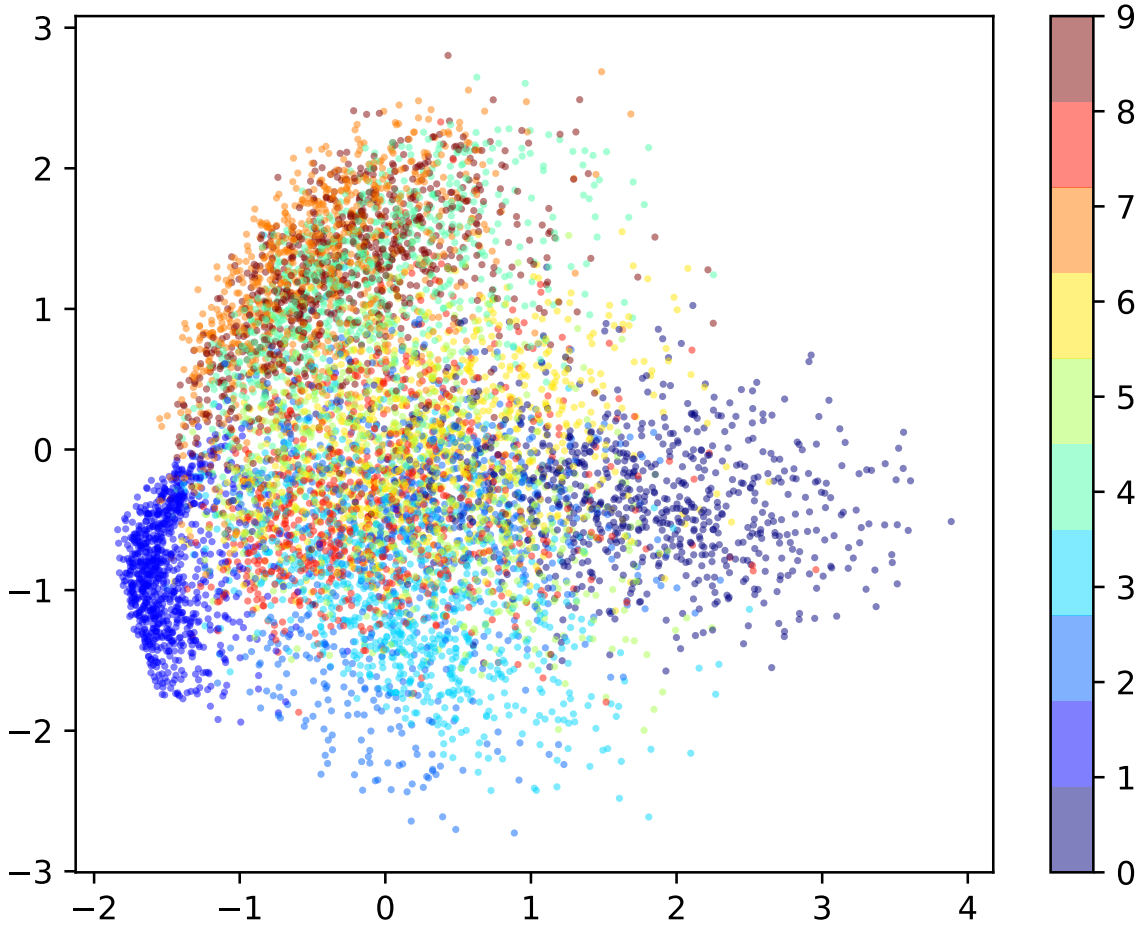


Figure courtesy of Matt Gormley

PCA Example: MNIST Digits

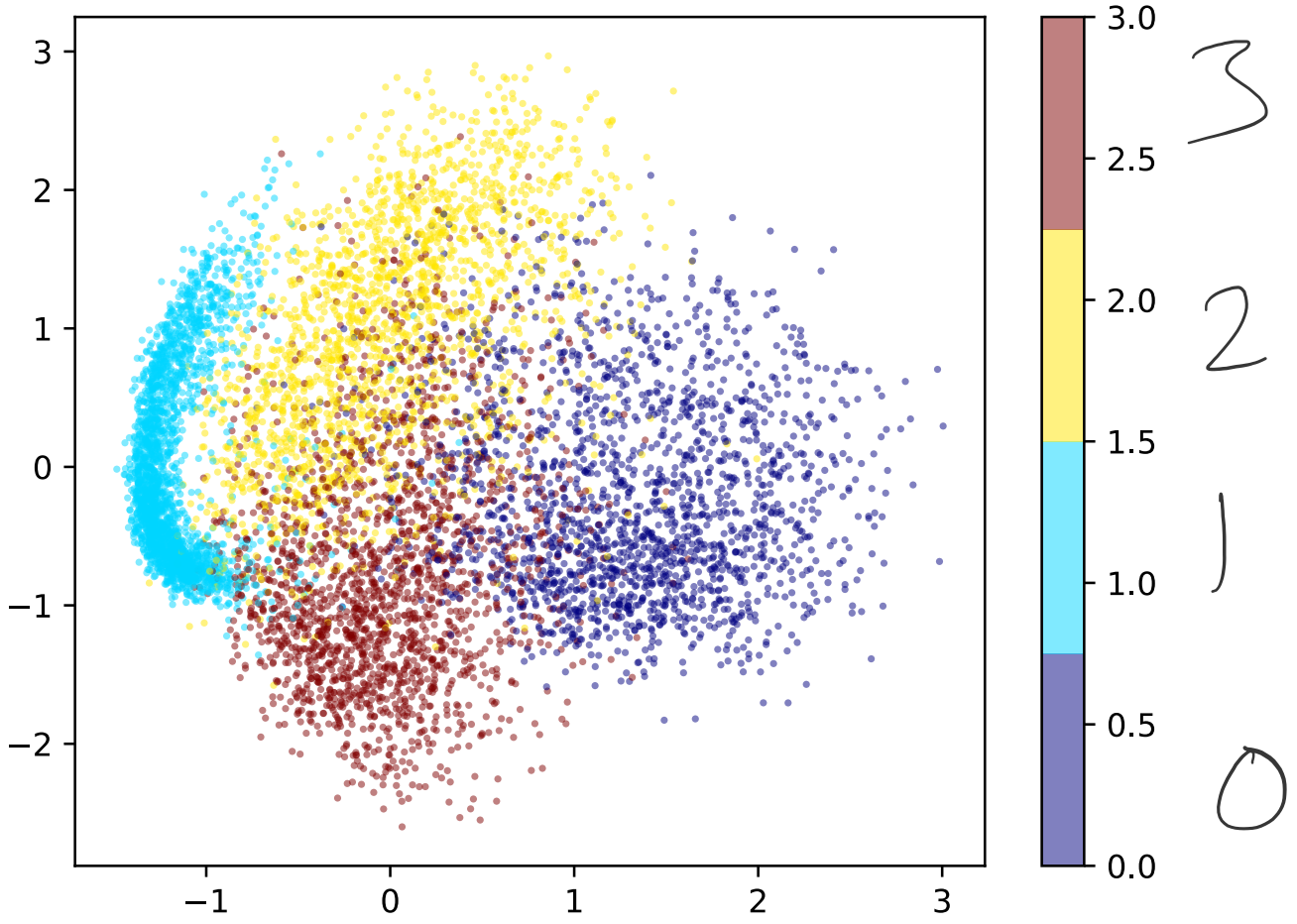
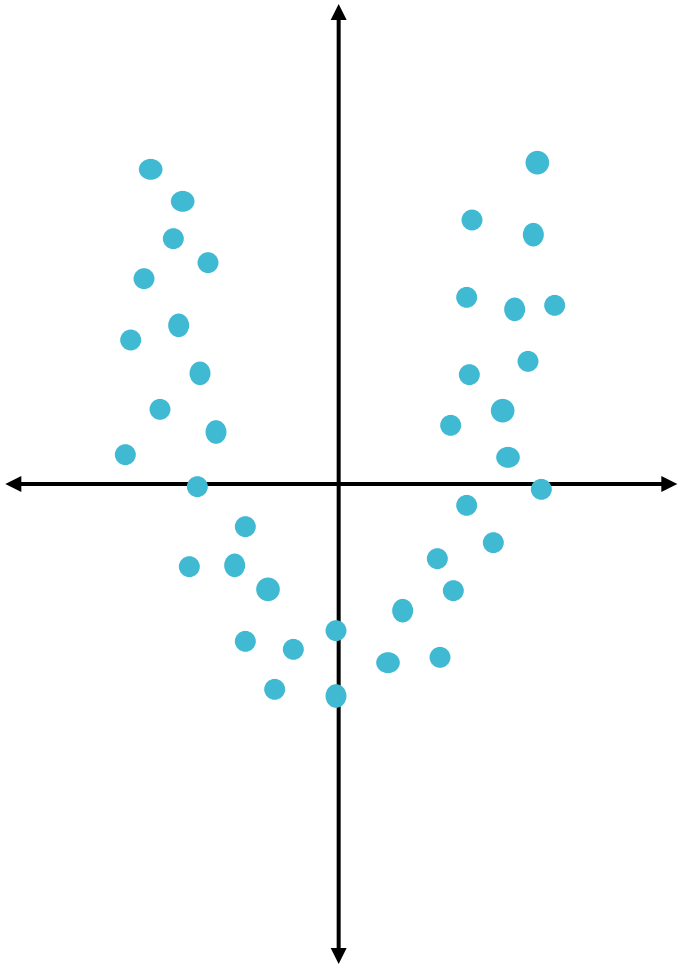


Figure courtesy of Matt Gormley

Shortcomings of PCA

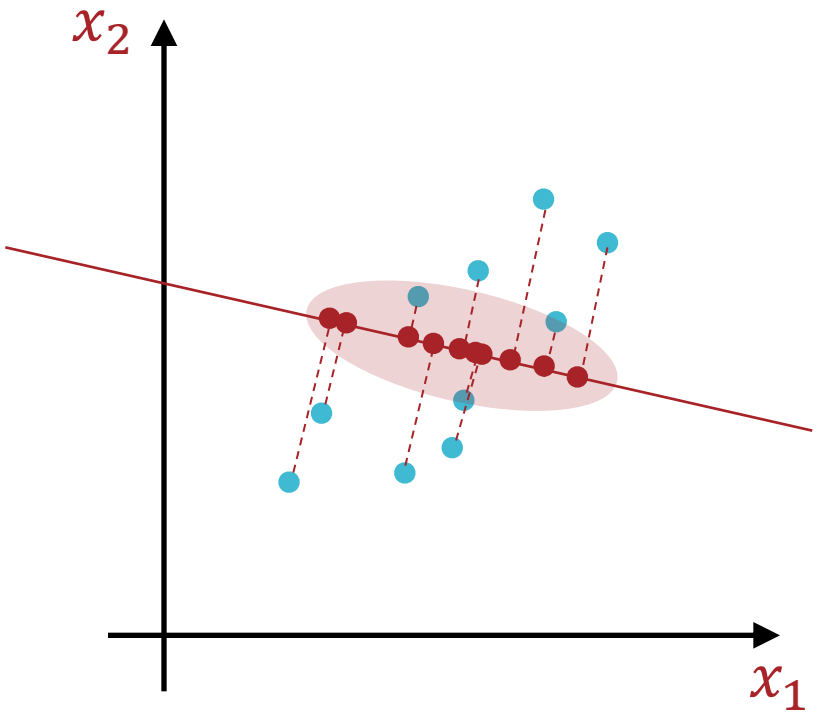


- Sometimes we don't care about variance
- PCA only finds linear combinations of our features
- Computationally intractable when k is very large
- interpretability
- othogonality constraint

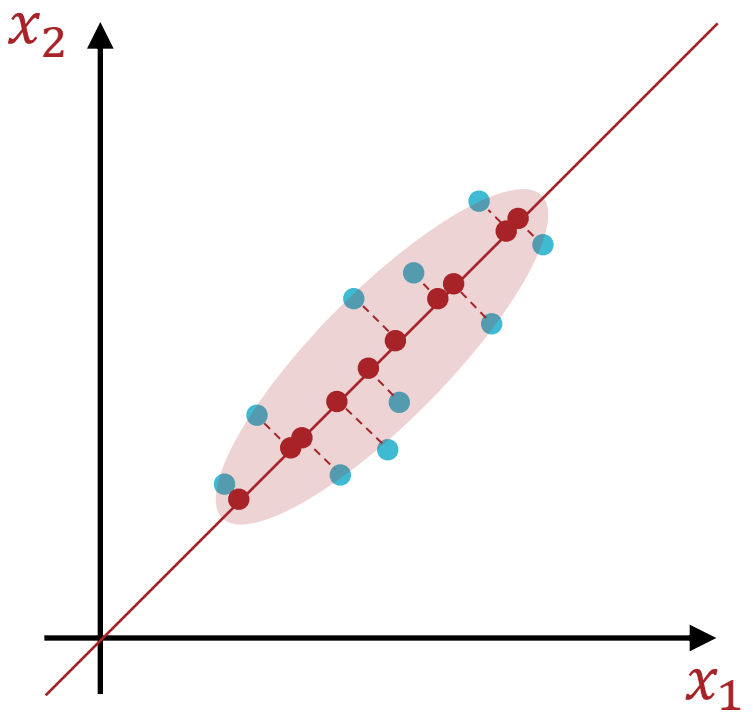
PCA Algorithm: Computational Cost

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. Center the data
   A. Optionally, normalize the data by features so that all features are of the same scale
2. Compute the covariance matrix $C_X = X^T X$ $(O(nk^2))$
3. Collect the top r eigenvectors (corresponding to the r largest eigenvalues), $P \in \mathbb{R}^{k \times r}$ $(O(k^3))$
4. Project the data into the space defined by P, $Z = XP$ $(O(nkr))$



Option A



Option B

Maybe Option A isn't so bad?

Random Projections?

- Issue: when k is very large, computing principal components can be intractable
  - PCA also requires the entire dataset to be available, which might not be possible (e.g., online learning)
- Idea: instead of rigorously minimizing the reconstruction error/maximizing the projections' variance, just pick a random set of vectors to project onto!
  - Each element of P is sampled from a standard Gaussian distribution, $P_{i,j} \sim N(0,1)$
  - Projections are still given by $Z = XP$
- Inquiry: is this a good idea? And if so, how good?

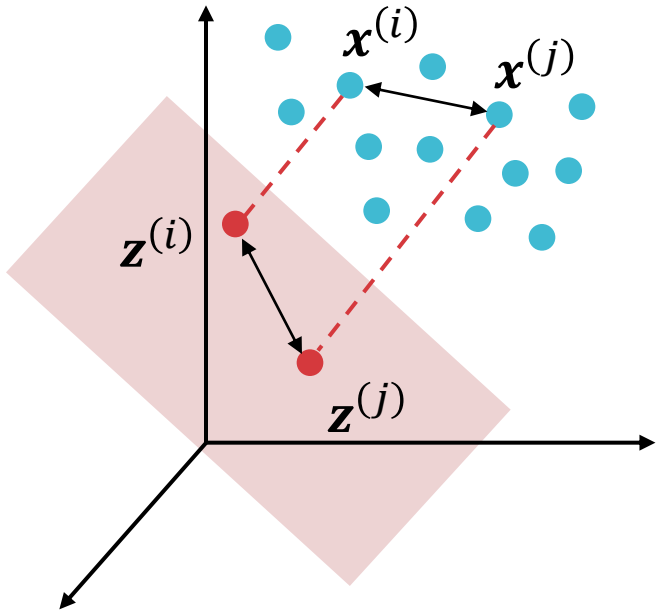
Random Projections: Computational Cost

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. ~~Center the data~~
   A. ~~Optionally, normalize the data by features so that all features are of the same scale~~
2. ~~Compute the covariance matrix $C_X = X^T X$ ($O(nk^2)$)~~
3. Generate P by sampling each element from a standard Gaussian distribution ($O(kr)$)
4. Project the data into the space defined by P, $Z = XP$ ($O(nkr)$)

Preserving Relative Distances

- Reasonable desideratum: distances between points are similar before and after being projected



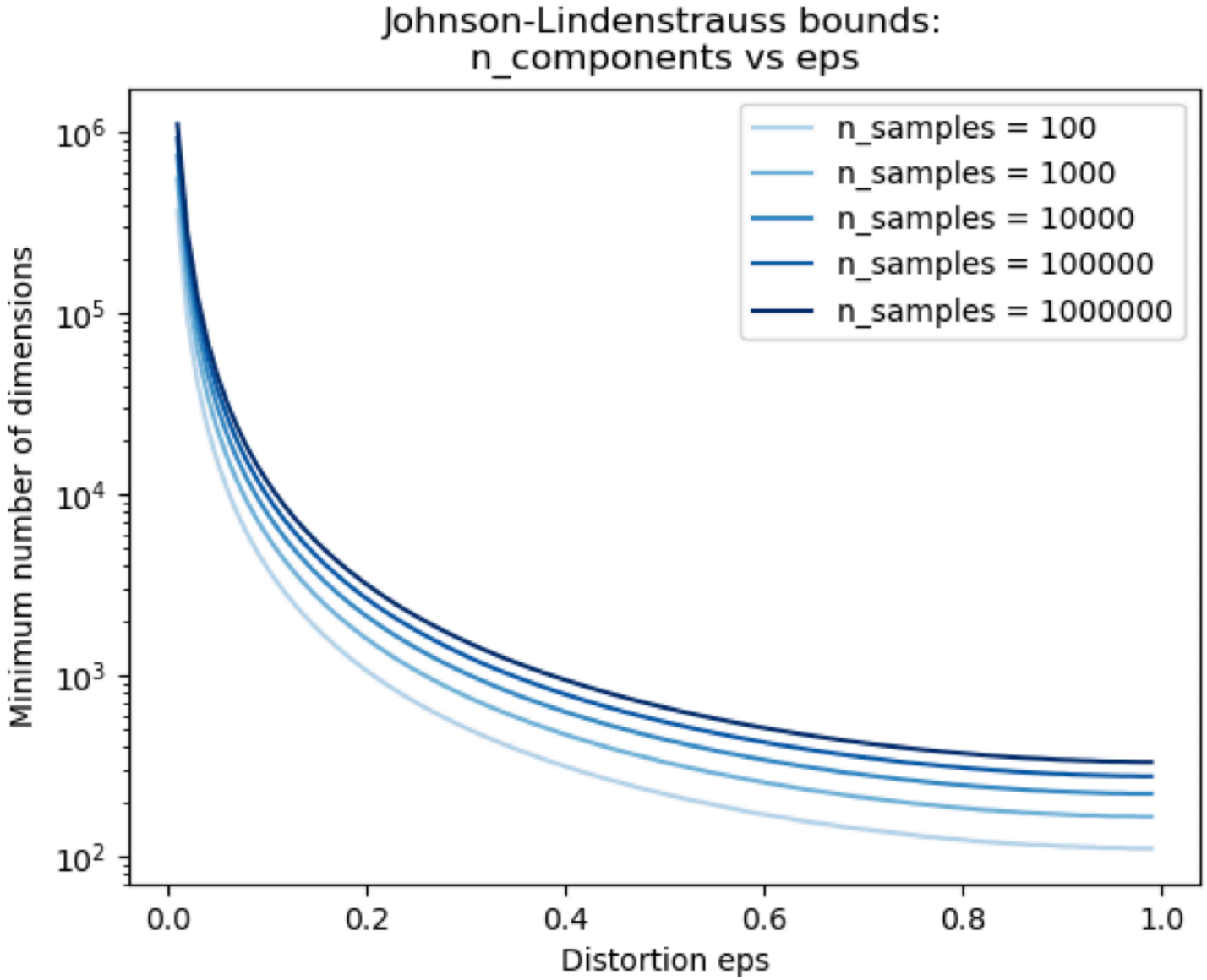
- Formally:

$$\left\|z^{(i)} - z^{(j)}\right\|_2^2 \in \left[(1-\epsilon)\left\|x^{(i)} - x^{(j)}\right\|_2^2, (1+\epsilon)\left\|x^{(i)} - x^{(j)}\right\|_2^2\right]$$

$$\forall\, x^{(i)}, x^{(j)} \in \mathcal{D}$$

Johnson-Lindenstrauss Lemma

Johnson-Lindenstrauss Lemma



Source: https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_johnson_lindenstrauss_bound.html

Johnson-Lindenstrauss Lemma: Proof (Sketch)

Key Takeaways

- Visualization is a key component of data pre-processing
  - Visualizing big data presents unique challenges
- PCA is dimensionality reduction technique that finds an orthonormal latent representation
  - Minimizes reconstruction error ↔ maximizing the projection variance
  - Computationally expensive (cubic in the number of features)
- Random projections are an efficient alternative
  - Johnson-Lindenstrauss lemma provides theoretical guarantee for preserving distances

Key Takeaways

- Visualization is a key component of data pre-processing
  - Visualizing big data presents unique challenges
- PCA is dimensionality reduction technique that finds an orthonormal latent representation
  - Minimizes reconstruction error ↔ maximizing the projection variance
  - Constrained to find projections that are linear combinations of the existing features
  - Computationally expensive (cubic in the number of features)