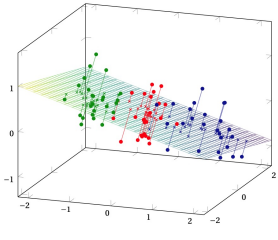
-Unsupervised learning
- PCA

Unsupervised learning: sample points but no labels
↳ no classes, no y-values, nothing to predict
↳ goal: discover structure of data

Examples of Unsupervised learning:
- **Clustering:** partition data into groups of similar/nearby pts.
- **dimensionality reduction:** data often lies on an underlying lower dimension
  - matrices have low rank approximations
- clustering was about finding similarity, dim. reduction more about finding variation
- **density estimation:** fit continuous distribution to discrete data
  - e.g. MLE to fit gaussians
  - can do more complicated functions

Principal Component Analysis: given pts. in $\mathbb{R}^d$, find k directions that capture most of the variance (dimensionality reduction)



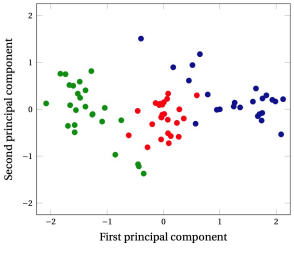


Figure 20.1: An example of PCA applied in a 3D dataset, with two principal components

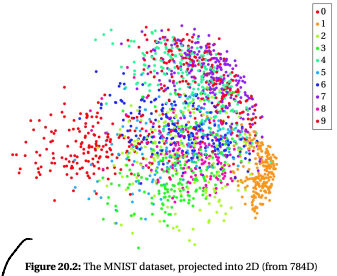


Figure 20.2: The MNIST dataset, projected into 2D (from 784D)

↳ 2 dimensions not enough to seperate data, (except 1 vs. 0)

Why PCA?
- dim. reduction → faster/cheaper compute
- removes irrelevant features/dimensions → less likely to overfit } preprocessing
- PCA is like subset selection except the features aren't axis aligned
  - instead, they're lin. combos of input features
- PCA finds small basis for representing variations in complex things

Computing the PCA:

$X \in \mathbb{R}^{n \times d}$ is the design matrix (no fictitious dim.)

- assume centered X, i.e. $\mu_X = 0$

Picking just 1 principal component:

Consider unit vector $\vec{w}$: orthogonal projection of $\vec{x}$ on $\vec{w}$: $\vec{x} = (\vec{x}^T \vec{w})\vec{w}$

if $\vec{w}$ doesn't have unit length: $\vec{x} = \frac{(\vec{x}^T \vec{w})}{\|\vec{w}\|^2} \vec{w}$

goal: pick best direction $\vec{w}$ and project all the data onto $\vec{w}$ to analyze data in 1-D
- can generalize to k-directions

The k vectors form our subspace/basis

if $\vec{v}_1 \dots \vec{v}_k$ orthogonal, then $\vec{x} = \sum_{i=1}^{k} (\vec{x}^T \vec{v}_i)\vec{v}_i$

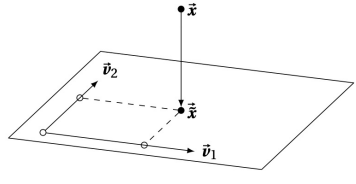


Figure 20.3: Example of projection onto a 2D subspace

Sometimes, we only want the principal coordinates $\vec{x}^T \vec{v}_i$ (don't actually project them)

Consider $X^T X$, which is square symmetric and PSD $\in \mathbb{R}^{d \times d}$

by Spectral thm, its eigenvalues are real $\rightarrow$ $0 \leq \lambda_1 \leq \dots \leq \lambda_d$

let $\vec{v}_1 \dots \vec{v}_d$ be the corresponding orthogonal unit eigenvectors (principal components)

greater eigenvalue $\rightarrow$ more important component/direction

Fitting Gaussians via MLE:

Let's fit a Gaussian to data w/ MLE

↳ Choose k Gaussian axes of greatest variance

Recall from MLE: estimate $\hat{\Sigma} = \frac{1}{n} X^T X$, X centered

→ PCA algo:

- Center X
- Normalize X (optional)
  - if units of measurement are different, then normalize (or else bad for Principal components)
  - otherwise, usually don't normalize
    - Want to maintain different variances (assuming same unit of measurement)
- Compute unit eigenvectors & eigenvalues of $X^T X$
- Optionally choose k based on eigenvalue sizes
- Choose k eigenvectors corresponding to k largest eigenvalues
- Compute principal coordinates $\vec{x}_i^T \vec{v}_i$ of training or test data
  - 2 choices

    1) Un-center input training data before projecting

    2) translate test data by same vector we used to translate training data when we centered it

maximizing Sample Variance

- 2nd derivation of PCA

goal: find $\vec{w}$ that maximizes sample variance of projected data

Rayleigh Quotient

formally: $\vec{w}^* = \underset{\vec{w}}{\operatorname{argmax}} \left(\operatorname{Var}\left(\{ \vec{x}_1 \cdots \vec{x}_n \}^T \right) = \frac{1}{n} \sum \left(x_i \cdot \frac{\vec{w}}{\|\vec{w}\|} \right)^2 = \frac{1}{n} \frac{\|X\vec{w}\|^2}{\|\vec{w}\|^2} = \frac{1}{n} \frac{\vec{w}^T X^T X \vec{w}}{\vec{w}^T \vec{w}} \right)$

$\rightarrow w^* = \underset{\vec{w}}{\operatorname{argmax}} \left(\frac{1}{n} \cdot \frac{\vec{w}^T X^T X \vec{w}}{\vec{w}^T \vec{v}} = \frac{\|X\vec{w}\|^2}{\|\vec{w}\|^2} \right) = \frac{\|x_i \vec{w}\|^2}{\|\vec{w}\|^2} = \lambda_i$

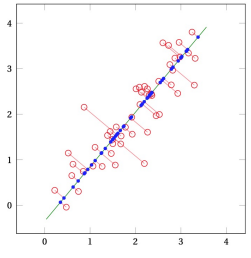


Figure 20.4: Points projected onto a line that maximizes the sample variance of the projected points

$\rightarrow$ eigenvector of λ_{max} maximizes variance $\frac{1}{n}\lambda_{max}$

One can show that this $\vec{v}_{max}$ beats the other vectors since each candidate $\vec{w}$ is a lin. combo of eigenvectors

↳ rayleigh quotient is convex combo. of eigenvalues

k directions $\rightarrow$ pick k largest $(\lambda, \vec{v})$ pairs that are mutually orthogonal

minimizing mean squared projection distance

3rd derivation: find $\vec{w}$ that minimizes the mean squared proj. distance

- analogous to LS regression but
  - w/ one subtle difference: error (sin) in fixed vertical direction but orthogonal direction instead

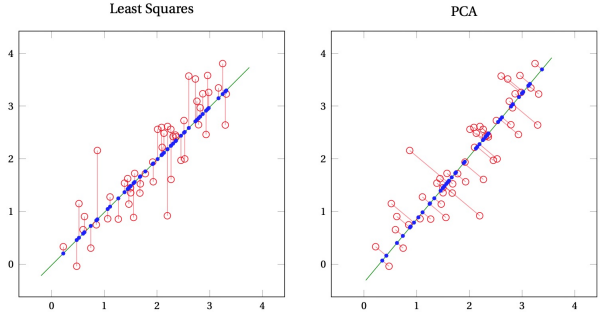


Figure 20.5: Comparison between least squares and PCA

→ find $\vec{u}$ that minimizes $\sum_{i=1}^{n} \| x_i - \tilde{x}_i \|^2 = \sum_{i=1}^{n} \| x_i - \frac{x_i \vec{u}}{\|\vec{u}\|} \cdot \vec{u} \|^2$

$= \sum_{i=1}^{n} \| x_i \|^2 - \sum_{i=1}^{n} \left(x_i \cdot \frac{\vec{u}}{\|\vec{u}\|} \right)^2$

(Constant) (n · variance from 2nd derivation)

⟹ Minimizing Mean-Squared projection distance ≡ Minimizing Variance

Eigenfaces: Use PCA for face recognition. $X \in \mathbb{R}^{n \times d}$, d pixels, n faces

200×200 image → 40000-length vector

goal: find nearest neighbor in $\mathbb{R}^d$ given image, compare w/ all training faces

issue: each query takes $\Theta(nd)$ time

↳ use PCA to reduce faces to smaller dim. $\mathbb{R}^{d'}$

→ now $\Theta(nd') \ll \Theta(nd)$