PCA

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Administrivia

- Problem set 4 (only coding) + 5 released today (only Math). Due on March 15.

Outline

Adaboost Algorithm

- Given: N samples $\{\boldsymbol{x}_n, y_n\}$, where $y_n \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- For $t = 1$ to T, choose a weak learner $h_t(\boldsymbol{x})$ and a contribution β_t.
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

Why AdaBoost works ?

Choosing the t-th classifier Suppose we have built a classifier $a_{t-1}(\boldsymbol{x})$, and we want to improve it by adding a weak learner $h_t(\boldsymbol{x})$

$$a(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

How can we choose the new classifier $h_t(\boldsymbol{x})$ and coefficient β_t? Adaboost greedily *minimizes the empirical risk of the exponential loss function*.

$$\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n e^{-y_n a(\boldsymbol{x}_n)} \\
&= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n e^{-y_n [a_{t-1}(\boldsymbol{x}_n) + \beta_t h_t(\boldsymbol{x}_n)]} \\
&= \arg\min_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)}
\end{aligned}$$

where we have used $w_t(n)$ as a shorthand for $e^{-y_n a_{t-1}(\boldsymbol{x}_n)}$ normalized to sum to 1 across the training examples n.

The new classifier

We can decompose the *weighted* loss function into two parts

$$
\begin{aligned}
&\sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)} \\
&= \sum_n w_t(n) e^{\beta_t} \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] + \sum_n w_t(n) e^{-\beta_t} \mathbb{I}[y_n = h_t(\boldsymbol{x}_n)] \\
&= \sum_n w_t(n) e^{\beta_t} \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] + \sum_n w_t(n) e^{-\beta_t} (1 - \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]) \\
&= (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] + e^{-\beta_t} \sum_n w_t(n)
\end{aligned}
$$

We have used the following properties to derive the above

- $y_n h_t(\boldsymbol{x}_n)$ is either 1 or -1 as $h_t(\boldsymbol{x}_n)$ is the output of a binary classifier
- The indicator function $\mathbb{I}[y_n = h_t(\boldsymbol{x}_n)]$ is either 0 or 1, so it equals $1 - \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]$

Finding the optimal weak learner

Summary

$$
\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \arg\min_{(h_t(\boldsymbol{x}), \beta_t)} \sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)} \\
&= \arg\min_{(h_t(\boldsymbol{x}), \beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] \\
&\qquad + e^{-\beta_t} \sum_n w_t(n)
\end{aligned}
$$

What term(s) must we optimize to choose $h_t(x_n)$?

$$
h_t^*(\boldsymbol{x}) = \arg\min_{h_t(\boldsymbol{x})} \epsilon_t = \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]
$$

Minimize weighted classification error as noted in step 1 of Adaboost!

How to choose β_t?

Summary

$$
\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \arg\min_{(h_t(\boldsymbol{x}), \beta_t)} \sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)} \\
&= \arg\min_{(h_t(\boldsymbol{x}), \beta_t)} (e^{\beta_t} - e^{-\beta_t}) \epsilon_t \\
&\qquad + e^{-\beta_t} \sum_n w_t(n)
\end{aligned}
$$

What term(s) must we optimize to choose β_t?

We need to minimize the entire objective function with respect to β_t!

We can do this by taking derivative with respect to β_t, setting to zero, and solving for β_t. After some calculation and using $\sum_n w_t(n) = 1$, we find:

$$
\beta_t^* = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}
$$

which is precisely step 2 of Adaboost! (*Exercise – verify the solution*)

Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$a_t(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

We then need to compute the new weights for each example:

$$\begin{aligned} w_{t+1}(n) &\propto e^{-y_n a_t(\boldsymbol{x}_n)} = e^{-y_n [a_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x}_n)]} \\ &= w_t(n) e^{-y_n \beta_t^* h_t^*(\boldsymbol{x}_n)} = \begin{cases} w_t(n) e^{\beta_t^*} & \text{if } y_n \neq h_t^*(\boldsymbol{x}_n) \\ w_t(n) e^{-\beta_t^*} & \text{if } y_n = h_t^*(\boldsymbol{x}_n) \end{cases} \end{aligned}$$

Intuition Misclassified data points will get their weights increased, while correctly classified data points will get their weight decreased

Adaboost Algorithm

- Given: N samples $\{\boldsymbol{x}_n, y_n\}$, where $y_n \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(n) = \frac{1}{N}$ for every training sample
- For $t = 1$ to T
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(n)$, by minimizing the weighted classification error

     $$\epsilon_t = \sum_n w_t(n)\mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

     $$w_{t+1}(n) \propto w_t(n)e^{-\beta_t y_n h_t(\boldsymbol{x}_n)}$$

     and normalize them such that $\sum_n w_{t+1}(n) = 1$
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T}\beta_t h_t(\boldsymbol{x})\right]$$

Meta-Algorithm

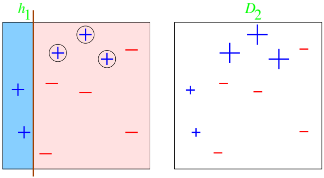
Note that the AdaBoost algorithm itself never specifies how we would get $h_t^*(\boldsymbol{x})$ as long as it minimizes the weighted classification error

$$\epsilon_t = \sum_n w_t(n)\mathbb{I}[y_n \neq h_t^*(\boldsymbol{x}_n)]$$

In this aspect, the AdaBoost algorithm is a meta-algorithm and can be used with any type of classifier

Decision Stumps

How do we choose the decision stump classifier given the weights at the second round of the following distribution?



We can simply enumerate all possible ways of putting vertical and horizontal lines to separate the data points into two classes and find the one with the smallest weighted classification error! Runtime?

- Presort data by each feature in O($dN \log N$) time
- Evaluate $N + 1$ thresholds for each feature at each round in O(dN) time
- In total O($dN \log N + dNT$) time – this efficiency is an attractive quality of boosting!

Outline

Raw data can be Complex, High-dimensional

To understand a phenomenon we measure various related quantities

If we knew what to measure or how to represent our measurements we might find simple relationships

But in practice we often *measure redundant signals*, e.g., US and European shoe sizes

Dimensionality Reduction

Issues

- *Measure redundant signals*

Goal: Find a 'better' representation for data

- To visualize and discover hidden patterns
- Preprocessing for supervised task

How do we define 'better'?

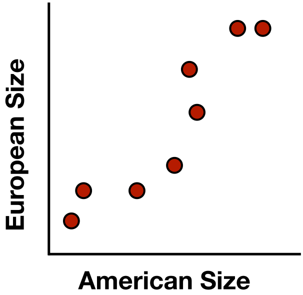
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



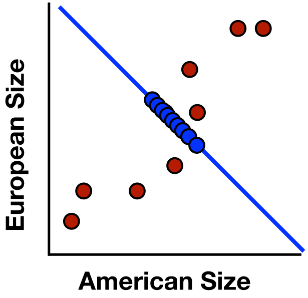
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



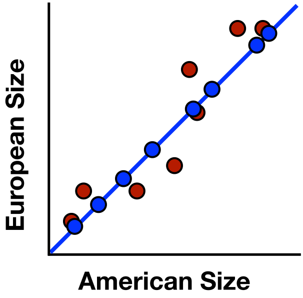
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

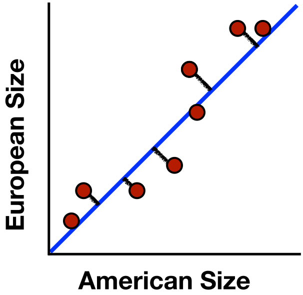
- Pick a direction and project onto this direction



Goal: Minimize Reconstruction Error

Minimize Euclidean distances between original points and their projections

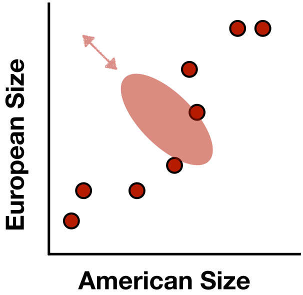
PCA solution solves this problem!



Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

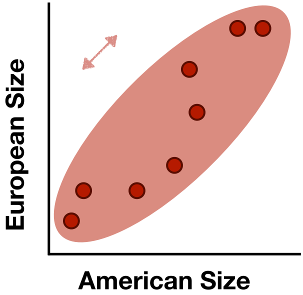
Can we do ‘better’, i.e., find a compact representation that captures variation?



Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

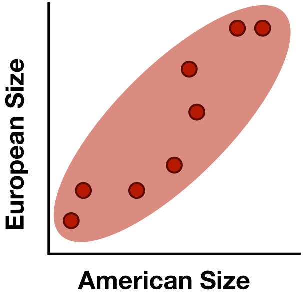


Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

PCA solution finds directions of maximal variance!



PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)

Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) simplifies problem

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{X} & \\ & & \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Variance of 1st feature $$\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)} - \mu_1\right)^2$$

Variance of 1st feature (assuming zero mean) $$\sigma_1^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_1^{(i)}\right)^2$$

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Covariance of 1st and 2nd features (assuming zero mean) $\sigma_{12} = \frac{1}{n} \sum_{i=1}^{n} x_1^{(i)} x_2^{(i)}$

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero → uncorrelated
- Large magnitude → (anti) correlated / redundant

Covariance Matrix

Covariance matrix generalizes this idea for many features

$d \times d$ covariance matrix with zero mean features

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$$

- ith diagonal entry equals variance of ith feature
- ijth entry is covariance between ith and jth features
- Symmetric (makes sense given definition of covariance)

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)

Find $\mathbf{P}$ such that the variance of the reduced representation $\mathbf{Z}$ is maximized.
Equivalent to finding $\mathbf{P}$ such that the reconstruction error of the raw data is minimized.

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA ‘scores’)
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)

$\mathbf{P}$ equals the top k eigenvectors of $\mathbf{C_X}$

$$\begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \quad \mathbf{X} \quad \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^\top \mathbf{b} = 0$ and $\mathbf{d}^\top \mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ $\mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ $\mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ $\mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$

Orthonormal vectors are orthogonal and have unit norm

- $\mathbf{a}$ are $\mathbf{b}$ are orthonormal, but $\mathbf{b}$ are $\mathbf{d}$ are not orthonormal

Eigendecomposition

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\mathbf{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

Eigenvector / Eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$ (unit norm)
- Example: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \implies$ eigenvector: $\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ eigenvalue: $\lambda = 1$

PCA Solution

All covariance matrices have an eigendecomposition

- $\mathbf{C}_\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\boldsymbol{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

The d eigenvectors are orthonormal directions of max variance

- Associated eigenvalues equal variance in these directions
- 1st eigenvector is direction of max variance (variance is λ_1)

Choosing k

How should we pick the dimension of the new representation?

Visualization: Pick top 2 or 3 dimensions for plotting purposes

Other analyses: Capture 'most' of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Can choose k such that we retain some fraction of the variance, e.g., 95%

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA

Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA

PCA results dependent on scaling of data

- Data is sometimes rescaled in practice before applying PCA

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)

$$\begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix}$$

PCA Formulation, $k = 1$

PCA: find one-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ (reduced representation, PCA 'scores')
- $\mathbf{p}$ is $d \times 1$ (columns are k principal components)

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(z^{(i)} \right)^2 = ||\mathbf{z}||_2^2$$

PCA Formulation, $k = 1$

PCA: find one-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ (reduced representation, PCA 'scores')
- $\mathbf{p}$ is $d \times 1$ (columns are k principal components)

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(z^{(i)} \right)^2 = ||\mathbf{z}||_2^2$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}||\mathbf{z}||_2^2$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma^2_{\mathbf{z}}$ where $||\mathbf{p}||_2 = 1$

Relationship between Euclidean distance and dot product

$$\sigma^2_{\mathbf{z}} = \frac{1}{n}||\mathbf{z}||_2^2$$
$$= \frac{1}{n}\mathbf{z}^\top \mathbf{z}$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma^2_{\mathbf{z}}$ where $||\mathbf{p}||_2 = 1$

$$\sigma^2_{\mathbf{z}} = \frac{1}{n}||\mathbf{z}||_2^2$$

Relationship between Euclidean distance and dot product

$$= \frac{1}{n}\mathbf{z}^\top \mathbf{z}$$

Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$

$$= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top (\mathbf{X}\mathbf{p})$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}||\mathbf{z}||_2^2$$

Relationship between Euclidean distance and dot product

$$= \frac{1}{n}\mathbf{z}^\top \mathbf{z}$$

Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$

$$= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top (\mathbf{X}\mathbf{p})$$

Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top \mathbf{X}^\top$; associativity of multiply

$$= \frac{1}{n}\mathbf{p}^\top \mathbf{X}^\top \mathbf{X}\mathbf{p}$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

	$\sigma_{\mathbf{z}}^2 = \frac{1}{n}\|\|\mathbf{z}\|\|_2^2$
Relationship between Euclidean distance and dot product	$= \frac{1}{n}\mathbf{z}^\top\mathbf{z}$
Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$	$= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top(\mathbf{X}\mathbf{p})$
Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top\mathbf{X}^\top$; associativity of multiply	$= \frac{1}{n}\mathbf{p}^\top\mathbf{X}^\top\mathbf{X}\mathbf{p}$
Definition: $\mathbf{C}_{\mathbf{X}} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$	$= \mathbf{p}^\top\mathbf{C}_{\mathbf{X}}\mathbf{p}$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

	$\sigma_{\mathbf{z}}^2 = \frac{1}{n}||\mathbf{z}||_2^2$
Relationship between Euclidean distance and dot product	$= \frac{1}{n}\mathbf{z}^\top \mathbf{z}$
Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$	$= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top(\mathbf{X}\mathbf{p})$
Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top \mathbf{X}^\top$; associativity of multiply	$= \frac{1}{n}\mathbf{p}^\top \mathbf{X}^\top \mathbf{X}\mathbf{p}$
Definition: $\mathbf{C}_{\mathbf{X}} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$	$= \mathbf{p}^\top \mathbf{C}_{\mathbf{X}} \mathbf{p}$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C}_{\mathbf{x}} \mathbf{p}$ where $||\mathbf{p}||_2 = 1$

Connection to Eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

Connection to Eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$
- But this is the expression we're optimizing, and thus maximal variance achieved when $\mathbf{p}$ is top eigenvector of $\mathbf{C_X}$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

Connection to Eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$
- But this is the expression we're optimizing, and thus maximal variance achieved when $\mathbf{p}$ is top eigenvector of $\mathbf{C_X}$

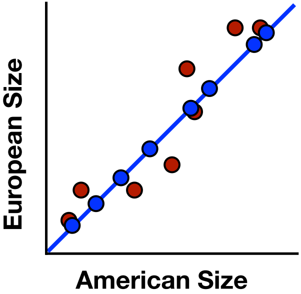
Similar arguments can be used for $k > 1$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

PCA Iterative Algorithm

$k = 1$: Find direction of max variance, project onto this direction

- Locations along this direction are the new 1D representation



PCA Iterative Algorithm

$k = 1$: Find direction of max variance, project onto this direction

- Locations along this direction are the new 1D representation

More generally, for i in $\{1, \ldots, k\}$:

- Find direction of max variance that is *orthonormal* to previously selected directions, project onto this direction
- Locations along this direction are the ith feature in new representation