



Dimensionality Reduction

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Eric Eaton (UPenn), Andrew Ng (Stanford), David Sontag (NYU), and the many others who made their course materials freely available online.

Robot Image Credit: Viktoriya Sukhanova © 123RF.com

Feature Selection

Learning Goals

- Describe the goal of feature selection
- Describe “simple” feature selection methods using greedy algorithms and L_1 -regularization

Setup

Problem

- new input data may have thousands or millions of dimensions

Dimensionality Reduction

- represent data with fewer dimensions

Why?



Feature Selection

Setup

- Want to learn $f: X \mapsto Y$
- But some features are more important than others

Problem

- Given d features, there are __ possible feature subsets
- Too expensive to explicitly compare all models

Approach

- Use a heuristic search procedure find a good feature subset
- Select subset of features to be used by learning algorithm
 - score each feature (or sets of features)
 - select set of features with best score



Greedy **Forward** Feature Selection Algorithm

- Pick dictionary of features
 - e.g. polynomials for linear regression
- Start from empty (or simple) set of features $\mathcal{F}_0 = \emptyset$
- For $t = 1$ to desired feature set size
 - Run learning algorithm for current set of features \mathcal{F}_t to obtain h_t
 - Select next best feature X_i , e.g. X_i that results in lowest test error when learning with $\mathcal{F}_t \cup \{X_i\}$
 - Set $\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t + \{X_i\}$
- Select and output the best feature subset that was evaluated during the entire search procedure

Greedy **Backward** Feature Selection Algorithm

- Pick dictionary of features
- Start with all features $\mathcal{F}_0 = \mathcal{F}$
- For $t = 1$ to desired feature set size
 - Run learning algorithm for current set of features \mathcal{F}_t to obtain h_t
 - Select next worst feature X_i , e.g. X_i that results in lowest test error when learning with $\mathcal{F}_i \setminus \{X_i\}$
 - Set $\mathcal{F}_{t+1} \leftarrow \mathcal{F}_t \setminus \{X_i\}$
- Select and output the best feature subset that was evaluated during the entire search procedure

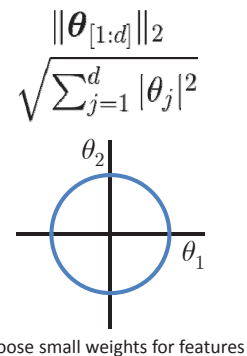
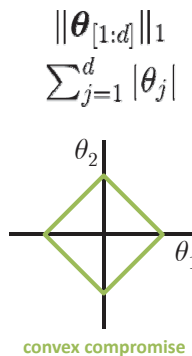
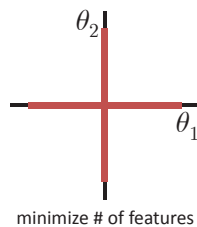
Feature Selection through **Regularization**

$$J_{n,\lambda}(\boldsymbol{\theta}) = \underbrace{R_n(\boldsymbol{\theta})}_{\text{empirical loss}} + \underbrace{\lambda z(\boldsymbol{\theta})}_{\text{regularization}}$$

What if we believe only a few features are relevant?

- L_1 -regularization $z(\boldsymbol{\theta}) = \|\boldsymbol{\theta}_{[1:d]}\|_1$
- Big area of ML called “sparse recovery”
- Recall

$$\|\boldsymbol{\theta}_{[1:d]}\|_0 = \sum_{j=1}^d |\theta_j|^0 = \sum_{j:\theta_j \neq 0} 1$$



Principal Component Analysis

Learning Goals

- Describe the goal of PCA
- Describe the relationship between PCA and eigenvectors
- Describe how to choose the number of PCs
- Describe uses (and misuses) of PCA

Dimensionality Reduction

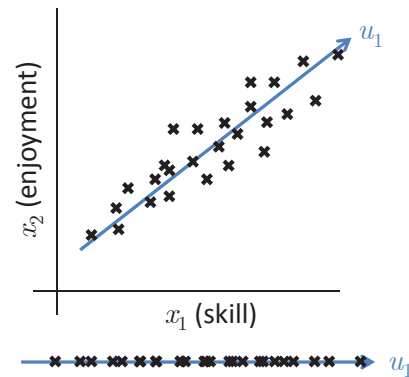
Assumption: data (approximately) lies on lower dimensional subspace

Example: survey of pilots for RC helicopters

$x_1^{(i)}$ = skill of pilot i

$x_2^{(i)}$ = enjoyment of pilot i

- Might expect that only the most committed students, the ones that truly enjoy flying, become good pilots
 - $x_1^{(i)}$ and $x_2^{(i)}$ are strongly correlated
 - Data actually lies along some diagonal axis u_1 that computes pilot “aptitude”, with small amount of noise lying off-axis



PCA: identify this subspace using eigenvectors

Based on notes by Andrew Ng

Lower Dimensional Projections

Rather than picking subset of features, obtain new ones by combining existing features X_1, \dots, X_d

$$z_1 = w_0^{(1)} + \sum_{j=1}^d w_j^{(1)} x_j$$

\vdots

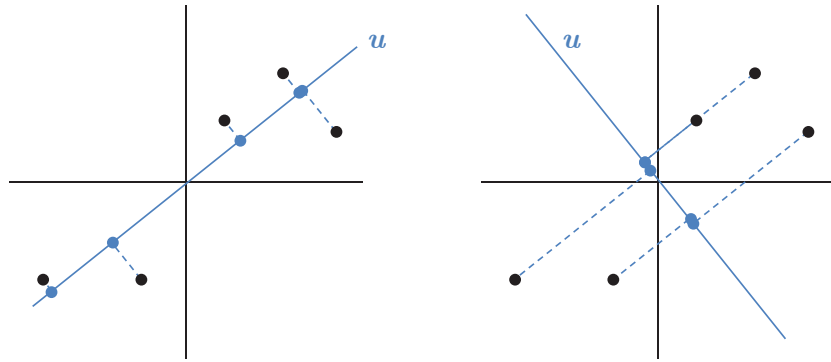
$$z_k = w_0^{(k)} + \sum_{j=1}^d w_j^{(k)} x_j$$

New features are linear combinations of old features

- Reduces dimension when $k < d$

We will consider how to do this in an unsupervised setting (just X , no Y)

Which Projection is Better?



Goal: automatically select direction u to

-
-

Based on notes by Andrew Ng



Review: Vector Projections

Basic definitions

$$\mathbf{x}^T \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

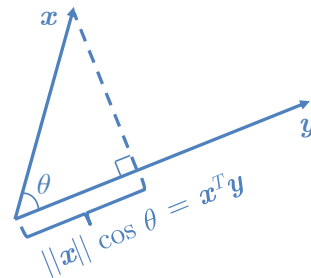
$$\cos \theta = \text{adj} / \text{hyp}$$

Then

$$\text{adj} = \text{hyp} \cos \theta$$

$$= \|\mathbf{x}\| \cos \theta$$

$$= \mathbf{x}^T \mathbf{y} \text{ [assuming } \|\mathbf{y}\| = 1 \text{ (unit vector)]}$$



Dot product is the length of the projection!

Formal Problem

For unit vector \mathbf{u} and point \mathbf{x} , the length of the projection of \mathbf{x} onto \mathbf{u} is $\mathbf{x}^T \mathbf{u}$

- For point $\mathbf{x}^{(i)}$, its projection onto \mathbf{u} is distance $(\mathbf{x}^{(i)})^T \mathbf{u}$ from the origin

To maximize the variance of the projections, choose unit-length \mathbf{u} so as to maximize

We will show that maximizing $\mathbf{u}^T \Sigma \mathbf{u}$ subject to $\|\mathbf{u}\|_2 = 1$ gives the **principal eigenvector** of Σ . That is, the first PC \mathbf{u} corresponds to the eigenvector of Σ with the largest eigenvalue.

Based on notes by Andrew Ng



Covariance Matrix Σ

Recall for single r.v. X ,

$$\sigma^2 = \text{var}(X) = \mathbb{E}[(X - \mu)^2] = \mathbb{E}[(X - \mu)(X - \mu)]$$

Let $\mathbf{X} = [X_1, \dots, X_d]^T$ for r.v.'s X_1, \dots, X_d .

(Note: \mathbf{X} is a vector of r.v.'s, not a matrix.)

Then $\Sigma_{ij} = \text{cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$

where $\mu_i = \mathbb{E}[X_i]$. Equivalently,

$$\begin{aligned} \Sigma &= \begin{bmatrix} \mathbb{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathbb{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \dots & \mathbb{E}[(X_1 - \mu_1)(X_d - \mu_d)] \\ \mathbb{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathbb{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \dots & \mathbb{E}[(X_2 - \mu_2)(X_d - \mu_d)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[(X_d - \mu_d)(X_1 - \mu_1)] & \mathbb{E}[(X_d - \mu_d)(X_2 - \mu_2)] & \dots & \mathbb{E}[(X_d - \mu_d)(X_d - \mu_d)] \end{bmatrix} \\ &= \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T] \end{aligned}$$

Based on notes by Andrew Ng

Review: Eigenvalues and Eigenvectors

Let \mathbf{A} be an $n \times n$ square matrix. Let λ be an eigenvalue of \mathbf{A} . Then $\exists \mathbf{v} \neq 0$ such that

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$

The vector \mathbf{v} is an **eigenvector** of \mathbf{A} associated with **eigenvalue** λ .

Notes

- The eigenvalues of \mathbf{A} are defined as the roots of
$$\text{determinant}(\mathbf{A} - \lambda\mathbf{I}) = |\mathbf{A} - \lambda\mathbf{I}| = 0.$$
- The vector \mathbf{v} points in a direction that is invariant under the associated linear transformation \mathbf{A} .
- There is no unique solution for \mathbf{v} . It is a direction vector only and can be scaled by any magnitude. Ordinarily, we normalize \mathbf{v} so that it has length one, that is, $\mathbf{v}^T\mathbf{v} = 1$.

Proof

$$\begin{aligned} & \max_{\mathbf{u}} \mathbf{u}^T \Sigma \mathbf{u} \\ & \text{s.t. } \mathbf{u}^T \mathbf{u} = 1 \end{aligned}$$



Summary

If we wish to find a 1D subspace with which to approximate the data (such that the variance of the projected data is maximized),

- Choose the principal component \mathbf{u} as the eigenvector of Σ corresponding to the largest eigenvalue of Σ

More generally, if we wish to project our data onto a k -dimensional subspace ($k < d$),

- Choose $\mathbf{u}_1, \dots, \mathbf{u}_k$ as the eigenvectors of Σ corresponding to the top k eigenvalues of Σ
- Since Σ is symmetric, the \mathbf{u}_j 's can be chosen to be orthogonal to each other so that the \mathbf{u}_j 's form a new, orthonormal basis for the data

Summary

To represent $\mathbf{x}^{(i)}$ in this new basis, we need only compute the corresponding vector

$$\mathbf{z}^{(i)} = \begin{bmatrix} \mathbf{u}_1^T (\mathbf{x}^{(i)} - \boldsymbol{\mu}) \\ \vdots \\ \mathbf{u}_k^T (\mathbf{x}^{(i)} - \boldsymbol{\mu}) \end{bmatrix}$$

Since PCA maps $\mathbf{x}^{(i)} \in \mathbb{R}^d$ to $\mathbf{z}^{(i)} \in \mathbb{R}^k$, where $\mathbf{z}^{(i)}$ is a lower, k -dimensional approximation of $\mathbf{x}^{(i)}$, PCA is a dimensionality reduction algorithm.

Minimal Approximation Error

PCA can also be derived by finding the basis that minimizes the approximation error arising from projecting the data onto the k -dimensional subspace spanned by them.

PCA Algorithm

Pre-process the data

- 1) Let $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)}$. zeros out mean of data
[may be omitted if data known to have zero mean]
Recenter the data by replacing $\mathbf{x}^{(i)}$ with $\mathbf{x}^{(i)} - \boldsymbol{\mu}$.

- 2) Let $\sigma_j^2 = \frac{1}{n} \sum_{i=1}^n (x_j^{(i)})^2$.
Normalize the data by replacing $x_j^{(i)}$ with $x_j^{(i)} / \sigma_j$.
rescales each coordinate to have unit variance,
ensuring all features are treated on the same "scale"
[may be omitted if different attributes known to be on same scale,
e.g. grayscale image with each feature $x_j \in [0, 255]$ as pixel intensity]

Run PCA

- 1) Compute covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} (\mathbf{x}^{(i)})^T$
- 2) Find eigenvalues and eigenvectors of $\boldsymbol{\Sigma}$.
- 3) Retain k eigenvectors with largest eigenvalues.

PCA Exercise

The covariance matrix corresponding to the standardized variables x_1 and x_2 is

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

Find its principal components (along with the associated weights).



(This slide intentionally left blank.)

Extensions: Matrix-Vector Notation

Let Σ be an $n \times n$ covariance matrix with eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_n$ and eigenvalues $\lambda_1, \dots, \lambda_n$, that is,

$$\Sigma \mathbf{u}_j = \lambda_j \mathbf{u}_j \text{ for } j = 1, \dots, n$$

Equivalently,

$$\begin{array}{ccc} \Sigma & \left[\begin{array}{c|c|c|c} | & | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_n \\ | & | & | & | \end{array} \right] & = & \left[\begin{array}{c|c|c|c} | & | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_n \\ | & | & | & | \end{array} \right] & \left[\begin{array}{cccc} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{array} \right] \\ \Sigma & \mathbf{U} & = & \mathbf{U} & \Lambda \end{array}$$

We normalized the eigenvectors to unit magnitude and they are orthogonal, so

$$\begin{aligned} \mathbf{U}\mathbf{U}^T &= \mathbf{U}^T\mathbf{U} = \mathbf{I} \\ \Rightarrow \Sigma\mathbf{U}\mathbf{U}^T &= \Sigma = \mathbf{U}\Lambda\mathbf{U}^T \end{aligned}$$

Extensions: Scaling Up

Covariance matrix Σ can be really big ($d \times d$)

Problems

- Finding eigenvectors is slow
- Computing Σ could have numerical precision issues

Use **singular value decomposition (SVD)**

For $n \times d$ matrix \mathbf{X} , there exists a factorization of the form

$$\mathbf{X} = \mathbf{W}\mathbf{S}\mathbf{V}^T$$

\mathbf{X} : $n \times d$ data matrix,
one row per data point
 \mathbf{W} : $n \times n$ singular vector matrix
 \mathbf{S} : $n \times d$ singular value matrix
 \mathbf{V} : $d \times d$ singular vector matrix

Then $\Sigma = \frac{1}{n}\mathbf{X}^T\mathbf{X}$

$$\begin{aligned} &= \frac{1}{n}(\mathbf{W}\mathbf{S}\mathbf{V}^T)^T(\mathbf{W}\mathbf{S}\mathbf{V}^T) \\ &= \frac{1}{n}\mathbf{V}\mathbf{S}\mathbf{W}^T\mathbf{W}\mathbf{S}\mathbf{V}^T \\ &= \frac{1}{n}\mathbf{V}\mathbf{S}^2\mathbf{V}^T = \mathbf{U}\Lambda\mathbf{U}^T \end{aligned}$$

(from previous slide)

- square roots of elements of \mathbf{S} are eigenvalues of Σ
- columns of \mathbf{V} are eigenvectors of Σ

- Retain k column vectors of \mathbf{V} with largest singular values

Choosing the Number of Principal Components

To choose k , note that

$$\text{average squared projected error} = \frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \right\|^2$$

$$\text{total variation in the data} = \frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} \right\|^2$$

Typically choose k to be the smallest value s.t. the error is upper-bounded.

e.g. To bound the error at 1%, choose k s.t.

$$\frac{\frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \right\|^2}{\frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} \right\|^2} \leq 0.01$$

This is equivalent to retaining 99% of the variance.

Algorithm

- 1) Try PCA with $k = 1$
- 2) Compute projected data
- 3) Check if ratio below threshold
- 4) Repeat

Based on slides by Andrew Ng

PCA Applications

Compression

- Speed up learning algorithm

– Training set: $\left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^n$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$

– Map inputs: $\left\{ \mathbf{x}^{(i)} \right\}_{i=1}^n \xrightarrow{\text{PCA}} \left\{ \mathbf{z}^{(i)} \right\}_{i=1}^n$

– New training set: $\left\{ \left(\mathbf{z}^{(i)}, y^{(i)} \right) \right\}_{i=1}^n$, where $\mathbf{z}^{(i)} \in \mathbb{R}^k$ ($k < d$)

– Note

- mapping $\mathbf{x}^{(i)} \rightarrow \mathbf{z}^{(i)}$ should be defined by running PCA only on training set
- this mapping can then be applied to examples $\mathbf{x}_{\text{CV}}^{(i)}$ and $\mathbf{x}_{\text{test}}^{(i)}$ in cross-validation and test sets

- Reduce memory / disk needed to store data

Based on slides by Andrew Ng

PCA Applications

Visualization

- Example: MNIST, 28 x 28 dimensional images



PCA Applications

Noise reduction

- High-dimensional data is assumed to lie on linear subspace
- Noise introduces small variability captured in PCs of lesser significance

original image



denoised image



Details

- Divide the original 372×492 image into 12×12 patches
- View each patch as 144d vector
- Reduce to 15d vector using PCA

PCA Misuse

Q: What about this design for an ML system?

- Get training data, run PCA, train classifier, test on test set

Q: What about using PCA to avoid overfitting?



Take-Aways

- Dimensionality reduction
 - What it is and why it is useful
- Simple feature selection
 - Greedy approach
 - L_1 -regularization as type of feature selection
- Principal Component Analysis (PCA)
 - One of the most commonly used and most powerful unsupervised learning algorithms
 - Maximize variance / minimize reconstruction error
 - Relationship to covariance matrix and eigenvectors
 - Choosing the number of PCs
 - Uses and misuses