Dimensionality Reduction and Other Unsupervised Learning Algorithms

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Motivation 1: Data Compression

Suppose we have collected a dataset containing many features; but, we have just plotted 2 of them.

Also, suppose that we don't know they are the length in cm and in inches

This is redundant and we can reduce data from 2D to 1D

Question: What does really 'reducing dimensions' mean?

Motivation 1: Data Compression

What does really 'reducing dimensions' mean?

Find a fitted line and
project the data to another axis.
Then, measure the position of
each example on that line.

z_1 is a new feature that specifies
the location of each of those points
on the green line.

Motivation 1: Data Compression

What does really 'reducing dimensions' mean?

$$x^{(1)} \in \mathbb{R}^2 \mapsto z^{(1)} \in \mathbb{R}$$

$$x^{(2)} \in \mathbb{R}^2 \mapsto z^{(2)} \in \mathbb{R}$$

$$\vdots$$

$$x^{(m)} \in \mathbb{R}^2 \mapsto z^{(m)} \in \mathbb{R}$$

That's, if we allow ourselves to approximate the original dataset by projecting the original examples onto the green line,
then we need only one real number to specify this point on the line.

Data Compression

Reduce data from 3D to 2D

$$i.e.\ z^{(i)} \in \mathbb{R}^2$$

$$i.e.\ z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

Question

- Suppose we apply dimensionality reduction to a dataset of m examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$. As a result of this, we will get out:

  (i) A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(k)}\}$ of k examples where $k \leq n$.

  (ii) A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(k)}\}$ of k examples where $k > n$.

  (iii) A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ of m examples where $z^{(i)} \in \mathbb{R}^k$ for some value of k and $k \leq n$.

  (iv) A lower dimensional dataset $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ of m examples where $z^{(i)} \in \mathbb{R}^k$ for some value of k and $k > n$.

Motivation 2: Data Visualization

	x_1	x_2	x_3	x_4	x_5	x_6	
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...	...	...	...	...	...	...	

(source: from wikipedia.org)

e.g. $x^{(i)} \in \mathbb{R}^{50}$

Motivation 2: Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

e.g. $z^{(i)} \in \mathbb{R}^2$

i.e. reduce data from 50D to 2D

Motivation 2: Data Visualization

Every country is represented by a point $z^{(i)} \in \mathbb{R}^2$

Question

- Suppose you have a dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ where $x^{(i)} \in \mathbb{R}^n$. In order to visualize it, we apply dimensionality reduction and get $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ where $z^{(i)} \in \mathbb{R}^k$ is k-dimensional. In a typical setting, which of the following would you expect to be true? Circle all the apply.

(i) $k > n$

(ii) $k \leq n$

(iii) $k \geq 4$

(iv) $k = 2$ or $k = 3$ (since we can plot 2D or 3D data but don't have ways to visualize higher dimensional data)

Principal Component Analysis (PCA)

PCA: Problem Formulation

$x_i \in \mathbb{R}^2$

Want to reduce 2D to 1D

i.e. find a good line in which the data can be projected onto

PCA: Problem Formulation

Intuition: Find a lower dimensional surface, *e.g.* the green line, in which the data can be projected onto. Also, the sum of squares of the red lines must be minimized !

PCA: Problem Formulation

Intuition: Find a lower dimensional surface, *e.g.* the green line, in which the data can be projected onto. Also, the sum of squares of the red lines must be minimized !

Remark: Feature scaling beforehand.

PCA: Problem Formulation

Intuition: Find a lower dimensional surface, *e.g.* the green line, in which the data can be projected onto. Also, the sum of squares of the red lines must be minimized !

Remark: Feature scaling beforehand.

PCA: Problem Formulation

Definition (Reduce from 2D to 1D):

Find a direction (a vector $u^{(i)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error

In this case, PCA should find $u^{(1)}$ for us !

Remark: Either PCA gives $u^{(1)}$ or $-u^{(1)}$, this does not matter !

PCA: Problem Formulation

Definition (Reduce from 2D to 1D):

Find a direction (a vector $u^{(i)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error

Definition (Reduce from *n*D to kD):

Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error

PCA: Problem Formulation

$3D \rightarrow 2D$

$K = 2$

Definition (Reduce from *n*D to ID):

Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error

PCA: Problem Formulation

$3D \rightarrow 2D$

$K = 2$

Definition (Reduce from *n*D to ID):

Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error

PCA: Problem Formulation

$3D \rightarrow 2D$

$K = 2$

Definition (Reduce from *n*D to ID):

Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error

PCA: Problem Formulation

$3D \rightarrow 2D$

$K = 2$

Definition (Reduce from *n*D to ID):

Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error

How is PCA related to linear regression?

PCA is NOT Linear Regression

Linear Regression

PCA

They are completely different algorithms !

Question

- Suppose you run PCA on the dataset below. Which of the following would be a reasonable vector $u^{(1)}$ onto which to project the data? (By convention, we choose $u^{(1)}$ so that $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$, the length of the vector $u^{(1)}$, equals 1).

(i) $u^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

(ii) $u^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

(iii) $u^{(1)} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

(iv) $u^{(1)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

PCA Algorithm

Data Preprocessing

Before doing PCA, you should always do the following data-preprocessing step

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (*aka.* feature scaling / mean normalization**):**

$$\left.\begin{array}{l} \mu_j := \dfrac{1}{m}\Sigma_{i=1}^{m} x_j^{(i)} \\ \text{Replace each } x_j^{(i)} \text{ with } x_j - \mu_j . \end{array}\right\} \text{To make each feature having zero-mean}$$

$$\left.\begin{array}{l} \text{If different features on different scales} \\ (e.g.,\ x_1 = \text{size of house},\ x_2 = \text{number of bedrooms}), \\ \text{scale feature to have comparable range of values.} \end{array}\right\} \frac{x_j - \mu_j}{S_j}$$

PCA Algorithm

Goal: Reduce data from n-dimensions to k-dimensions

1 Compute 'covariance matrix':

$$\Sigma = \frac{1}{m}\Sigma_{i=1}^{n}\underbrace{(x^{(i)})}_{n\times 1}\underbrace{(x^{(i)})^T}_{1\times n}$$

$n \times n$

2 Compute 'eigenvector' of matrix Σ:
(*e.g.* by calling the function 'singular value decomposition' or **svd**)

$$(U, S, V) = \mathbf{svd}(\Sigma)$$

Note:

$$U = \begin{bmatrix} | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & u^{(2)} & u^{(3)} & \dots & u^{(m)} \\ | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \end{bmatrix} \text{ such that } U \in \mathbb{R}^{n\times n}$$

PCA Algorithm

Goal: Reduce data from n-dimensions to k-dimensions

1 Compute 'covariance matrix':

$$\Sigma = \frac{1}{m}\Sigma_{i=1}^{n}\underbrace{(x^{(i)})}_{n\times 1}\underbrace{(x^{(i)})^T}_{1\times n}$$

$n \times n$

2 Compute 'eigenvector' of matrix Σ:

(*e.g.* by calling the function 'singular value decomposition' or **svd**)

$$(U, S, V) = \mathbf{svd}(\Sigma)$$

Note:

$$U = \underbrace{\begin{bmatrix} | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & u^{(2)} & u^{(3)} & \dots & u^{(m)} \\ | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \end{bmatrix}}_{k}$$

such that $U \in \mathbb{R}^{n\times n}$

Pick the first k !

i.e.

$u^{(1)}, \dots, u^{(k)}$

PCA Algorithm

Goal: Reduce data from n-dimensions to k-dimensions

i.e. $X \in \mathbb{R}^n \mapsto z \in \mathbb{R}^k$

sometimes called 'U-reduced'

Solution:

$$z = \left(\begin{bmatrix} | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \\ u^{(1)} & u^{(2)} & u^{(2)} & u^{(3)} & \dots & u^{(k)} \\ | & | & | & | & \dots & | \\ | & | & | & | & \dots & | \end{bmatrix}_{n \times k}^{T} \right) \left(X_{n \times 1} \right)$$

$$= \left(\begin{bmatrix} - & - & (u^{(1)})^T & - & - \\ & & \vdots & - & - \\ - & - & (u^{(k)})^T & - & - \end{bmatrix}_{k \times n} \right) \left(X_{n \times 1} \right) \qquad \therefore z \in \mathbb{R}^k$$

PCA Algorithm

Goal: Reduce data from n-dimensions to k-dimensions

i.e. $X \in \mathbb{R}^n \mapsto z \in \mathbb{R}^k$

sometimes called 'U-reduced'

Solution:

$$z^{(i)} = \left(\begin{bmatrix} | & | & | & | & \cdots & | \\ | & | & | & | & \cdots & | \\ u^{(1)} & u^{(2)} & u^{(2)} & u^{(3)} & \cdots & u^{(k)} \\ | & | & | & | & \cdots & | \\ | & | & | & | & \cdots & | \end{bmatrix}_{n \times k}^{T} \right) \left(X^{(i)}_{n \times 1} \right)$$

$$= \left(\begin{bmatrix} - & - & (u^{(1)})^T & - & - \\ & & \vdots & - & - \\ - & - & (u^{(k)})^T & - & - \end{bmatrix}_{k \times n} \right) \left(X^{(i)}_{n \times 1} \right) \qquad \therefore z^{(i)} \in \mathbb{R}^k$$

PCA Algorithm (Vectorizing)

Goal: Reduce data from n-dimensions to k-dimensions

i.e. $X \in \mathbb{R}^n \mapsto z \in \mathbb{R}^k$

$$\because X = \begin{bmatrix} - & - & (x^{(i)})^T & - & - \\ - & - & \vdots & - & - \\ - & - & (x^{(m)})^T & - & - \end{bmatrix}$$

$$\therefore \Sigma = \frac{1}{m} X^T X$$

1 Compute 'covariance matrix':

$$\Sigma = \frac{1}{m} \Sigma_{i=1}^{n} \underbrace{(x^{(i)})}_{n \times 1} \underbrace{(x^{(i)})^T}_{1 \times n}$$

$n \times n$

2 Compute 'eigenvector' of matrix Σ:

(*e.g.* by calling the function 'singular value decomposition' or **svd**)

$(U, S, V) = \mathbf{svd}(\Sigma)$

3 $U_{\textbf{reduce}} := U(:, 1:k)$

Note: $X \in \mathbb{R}^n$ (not $X \in \mathbb{R}^{n+1}$)

4 $z := \left(U_{\textbf{reduce}}^T\right)\left(X\right)$

Question

- In PCA, we obtain $z \in \mathbb{R}^k$ from $x \in \mathbb{R}^n$ as follows:

$$z = \left(\begin{bmatrix} | & | & \dots & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & \dots & | \end{bmatrix}^T \right) \left(x \right) = \left(\begin{bmatrix} - & - & (u^{(1)})^T & - & - \\ - & - & (u^{(2)})^T & - & - \\ - & - & \vdots & - & - \\ - & - & (u^{(k)})^T & - & - \end{bmatrix} \right) \left(x \right)$$

Which of the following is a correct expression for z_j ?

(i) $z_j = (u^{(k)})^T x$

(ii) $z_j = (u^{(j)})^T x_j$

(iii) $z_j = (u^{(j)})^T x_k$

(iv) $z_j = (u^{(j)})^T x$

Reconstruction from Compressed Representation

Intuition

Question: If PCA is a compression algorithm,
there should be a way to de-compress a representation to its original ones ?
(an approximated original representation, probably ?)

$$z = \left(U^T_{\textbf{reduce}}\right)\left(X\right)$$

Intuition

Question: If PCA is a compression algorithm,
there should be a way to de-compress a representation to its original ones ?
(an approximated original representation, probably ?)

$$z = \left(U^T_{\textbf{reduce}}\right)\left(X\right)$$

Now, we want $z \in \mathbb{R}^k \mapsto X \in \mathbb{R}^n$

$$X_{\textbf{approx}} = \underbrace{\left(U_{\textbf{reduce}}\right)}_{n \times k}\underbrace{\left(z\right)}_{k \times 1}$$

$$\therefore X_{\textbf{approx}} \in \mathbb{R}^n$$

Intuition

Question: If PCA is a compression algorithm,
there should be a way to de-compress a representation to its original ones ?
(an approximated original representation, probably ?)

$$z = \left(U^T_{\textbf{reduce}}\right)\left(X\right)$$

Now, we want $z \in \mathbb{R}^k \mapsto X \in \mathbb{R}^n$

$$X_{\textbf{approx}} = \underbrace{\left(U_{\textbf{reduce}}\right)}_{n \times k}\underbrace{\left(z\right)}_{k \times 1}$$

$$\therefore X_{\textbf{approx}} \in \mathbb{R}^n$$

Question

- Suppose we run PCA with $k = n$, so that the dimension of the data is not reduced at all (This is not useful in practice but it is a good thought exercise). Recall that the percent / fraction of variance retained is given by $\frac{\Sigma_{i=1}^{k} S_{ii}}{\Sigma_{i=1}^{n} S_{ii}}$. Which of the following will be true? Circle all that apply.

  (i) $U_{\textbf{reduce}}$ will be an $n \times n$ matrix.

  (ii) $x_{\textbf{approx}} = x$ for every x.

  (iii) The percentage of variance retained will be 100%

  (iv) We have that $\frac{\Sigma_{i=1}^{k} S_{ii}}{\Sigma_{i=1}^{n} S_{ii}} > 1$.

Choosing the number of principal components

How to choose k

Criteria of choosing k (*i.e.* the number of principal components)

- Average squared projection error: $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2$
- Total variation in the data: $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2$

Typically, choose k to be smallest value so that:

$$\frac{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2}{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01$$

***i.e.* 99% of variance is retained !**

How to choose k

Criteria of choosing k (*i.e.* the number of principal components)

- Average squared projection error: $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2$
- Total variation in the data: $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2$

Typically, choose k to be smallest value so that:

$$\frac{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2}{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2} \leq \cancel{0.01}$$

0.05
0.10

***i.e.* ~~99~~% of variance is retained !**

95%
90%

Algorithm: How to choose k

Input: none
Output: k

Set k equal to 1
Repeat until the condition is satisfied
Compute $U_{\textbf{reduce}}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{\textbf{approx}}, \ldots, x^{(m)}_{\textbf{approx}}$

Check if the condition is satisfied *i.e.*

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01\,?$$

Otherwise, increment k

Algorithm: How to choose k

Input: none
Output: k

Set k equal to 1
Repeat until the condition is satisfied
Compute $U_{\textbf{reduce}}, z^{(1)}, z^{(2)}, \ldots,$
$z^{(m)}, x^{(1)}_{\textbf{approx}}, \ldots, x^{(m)}_{\textbf{approx}}$

Check if the condition is satisfied *i.e.*

$$\boxed{\frac{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2}{\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2}} \leq 0.01\,?$$

Otherwise, increment k

(U, S, V) = **svd(Σ)**

$$S = \begin{bmatrix} S_{11} & 0 & 0 & 0 & 0 \\ 0 & S_{22} & 0 & 0 & 0 \\ 0 & 0 & S_{33} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & S_{nn} \end{bmatrix}$$

For given k,

$$= 1 - \frac{\Sigma_{i=1}^{k} S_{ii}}{\Sigma_{i=1}^{n} S_{ii}}$$

Algorithm: How to choose k

Input: none
Output: k

Set k equal to 1
Repeat until the condition is satisfied
Compute $U_{\textbf{reduce}}, z^{(1)}, z^{(2)}, \ldots,$
$z^{(m)}, x^{(1)}_{\textbf{approx}}, \ldots, x^{(m)}_{\textbf{approx}}$

Check if the condition is satisfied *i.e.*

$$\boxed{\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}} \leq 0.01\,?$$

Otherwise, increment k

(U, S, V) = **svd(Σ)**

$$S = \begin{bmatrix} S_{11} & 0 & 0 & 0 & 0 \\ 0 & S_{22} & 0 & 0 & 0 \\ 0 & 0 & S_{33} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & S_{nn} \end{bmatrix}$$

For given k,

$$= 1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$$

Following this, we don't need to run PCA from scratch over and over

Question

- Previously, we said that PCA chooses a direction $u^{(1)}$ (or k directions $u^{(1)}, \ldots, u^{(k)}$) onto which to project the data so as to minimize the (squared) projection error. Another way to say the same is that PCA tries to minimize:

(i) $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}\|^2$

(ii) $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)}_{\textbf{approx}}\|^2$

(iii) $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} - x^{(i)}_{\textbf{approx}}\|^2$

(iv) $\frac{1}{m}\Sigma_{i=1}^{m}\|x^{(i)} + x^{(i)}_{\textbf{approx}}\|^2$

Advice for Applying PCA

PCA for Speed-up Learning

Dataset for supervised learning:

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}) \text{ , where } x^{(i)} \in \mathbb{R}^{30{,}000}$$

100 px

100 px

1. Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{30{,}000}$

$$\Big\downarrow \textbf{PCA}$$

$$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{3{,}000}$$

2. New training dataset:

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$$

PCA for Speed-up Learning

1. Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10{,}000}$

$$\Big\downarrow \textbf{PCA}$$

$$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1{,}000}$$

2. New training dataset:

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$$

Given a new *x*, we now do: $\mathcal{X} \longrightarrow \mathcal{Z}$

(assume that we are using logistic regression model)

$$h_\theta(z) = \frac{1}{1 + e^{-\theta^T z}}$$

***i.e.* running PCA on the training set !**

Application of PCA

- Compression
  - Reduce memory / disk needed to store data
  - Speed up learning algorithm

  } Choose k based on the percentage of variance retained

- Visualization

  → Since we can plot only on 2D or 3D data, then we often set $k = 2$ or $k = 3$!

Bad Use of PCA

1. Using PCA to prevent overfitting:

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

Since the number of features is reduced, then less likely to overfit !

This might work OK, but is not a good way to address overfitting.
Use regularization instead *i.e.*

$$\min_{\theta} \frac{1}{2m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \Sigma_{j=1}^{n} \theta_j^2$$

Why is this method not a good way to address overfitting?
(some information is loss !)

Bad Use of PCA

2. Just insist to do PCA without supportive reasons

Following is a design of machine learning system.

- Get a training dataset: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce dimensions of $x^{(i)}$ and obtain $z^{(i)}$
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set *i.e.* map $x^{(i)}_{\textbf{test}}$ to $z^{(i)}_{\textbf{test}}$
- Run $h_\theta(z)$ on $\{(z^{(1)}_{\textbf{test}}, y^{(1)}), (z^{(2)}_{\textbf{test}}, y^{(2)}), \ldots, (z^{(m)}_{\textbf{test}}, y^{(m)})\}$

How about doing the whole thing without using PCA?

Only your learning algorithm runs too slow;
or memory requirement and disk space requirement are too large.

Question

- Which of the following are good / recommended applications of PCA? Circle all that apply.

(i) To compress the data so that it takes up less computer memory / disk space

(ii) To reduce the dimension of the input data so as to speed up a learning algorithm

(iii) Instead of using regularization, use PCA to reduce the number of features to reduce overfitting

(iv) To visualize high-dimensional data (by choosing $k = 2$ or $k = 3$)

PCA *vs.* LDA

PCA *vs.* LDA

Principal Component Analysis	**Linear Discriminant Analysis**
Unsupervised learning	Supervised learning
Looks for properties that show as much variation across datasets as possible (*i.e.* maximize the variance !)	Find boundaries around clusters of data, which maximizes the distance between each centroid and minimizes the variation within each cluster
Better if #examples per class is less	Better if large dataset having multiple classes

If you are interested in an empirical comparison, see (published in 2001**):**

PCA versus LDA

Aleix M. Martínez and Avinash C. Kak

Robot Vision Lab
School of Electrical and Computer Engineering
Purdue University, IN 47907-1285
{aleix, kak}@ecn.purdue.edu

In the context of the appearance-based paradigm for object recognition, it is generally believed that algorithms based on LDA (Linear Discriminant Analysis) are superior to those based on PCA (Principal Components Analysis). In this communication we show that this is not always the case. We present our case first by using intuitively plausible arguments and then by showing actual results on a face database. Our overall conclusion is that when the training dataset is small, PCA can outperform LDA, and also that PCA is less sensitive to different training datasets.

Other Unsupervised Learning Algorithms

Other Learning Algorithms

In the brief overview, we have seen K-means for clustering and PCA for dimensionality reduction.

What else is out there? Most unsupervised learning algorithms have one or more of these aims:

1. Clustering, for purposes of discretizing or detecting anomalies,
2. Probability density estimation, for purposes of positive detection, anomaly detection, or synthesis of new data distributed similarly to the training set.
3. Latent space discovery, for purposes of dimensionality reduction, positive detection, synthesis of new data distributed similarly to the training set.

Clustering

We already saw how K-means can perform data clustering.

This style of clustering is sometimes called 'vector quantization'. The goal is coding of inputs as members of a discrete set according to spatial locality.

A second major group of clustering algorithms are pairwise clustering methods that use similarity distance to group inputs. Two types are:

- Hierarchical clustering is top-down
- Agglomerative clustering is bottom-up

Density Estimation

Density estimators can be used for detection (any input whose probability density is larger than some value is classified as a positive).

Density estimators can be used for anomaly detection (any input whose probability density is below some value is classified as anomalous).

(a) Classification (b) Anomaly detection

Latent Space Discovery

Latent space methods map the input space to a lower-dimensional representation, called a latent or semantic representation.

Principle components analysis (PCA) models the latent space as Gaussian distribution in the k-dimensional linear subspace of the original space in which the input data have the most variance.

Locally-linear embedding (LLE) and other nonlinear dimensionality reduction methods model the data as generated from a non-linear submanifold of the input space.

Latent Dirichlet allocation (LDA) is a topic model in NLP that imposes a prior on the data distribution in the latent space.

Latent Space Discovery

Latent space methods map the input space to a lower-dimensional representation, called a latent or semantic representation.

Generative adversarial networks (GANs) represent the mapping from the latent space to the data space by a neural network, usually a deconvolutional neural network. This generative model is trained alongside a discriminative adversary.

(Image from: https://medium.com/archieai/a-dozen-times-artificial-intelligence-startled-the-world-eae5005153db)

Summary

Unsupervised learning is an extremely rich area. It can be deserved an entire course on its own !