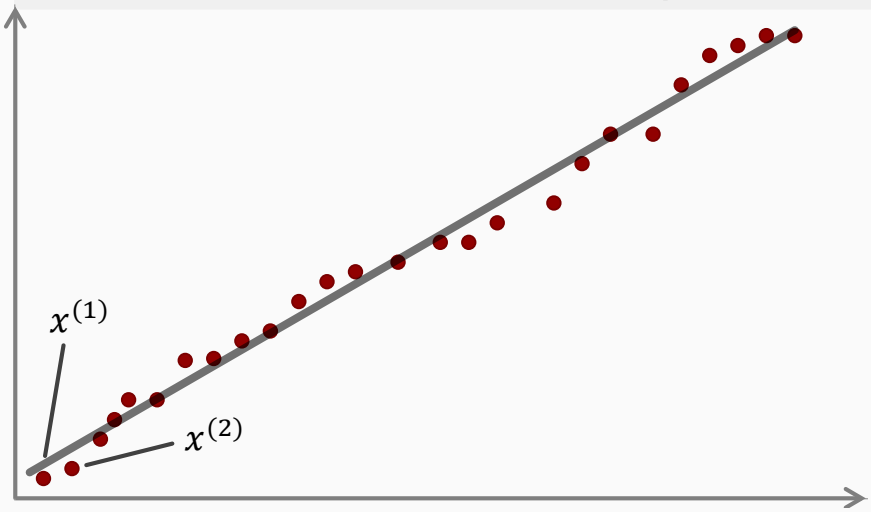
dimensionality reduction 🗜 principal components

motivation i & ii: data compression and visualization



two features in a dataset that are highly correlated (multicollinearity) create redundancy in the data and do not add predictive power to models considering the correlated variables represent essentially similar data.

(e.g. a unit in $ dollars or € euros)

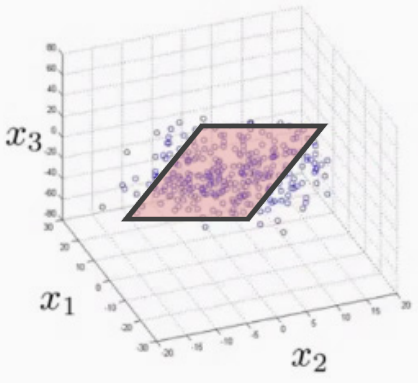
reducing the data from 2D to 1D:

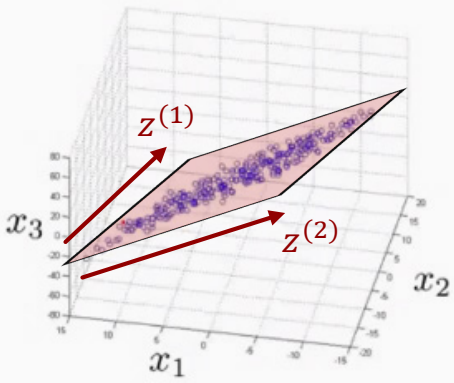
projecting all of the correlated data points on a straight line will provide a basis to measure the distance between each example. a new feature can be created ($z^{(1)}$) that specifies the location of each point on the original linear trend. only one number from each of the plotted examples will need to be retained to meet linear requirements. this will in effect, optimize algorithms for computational resources required.

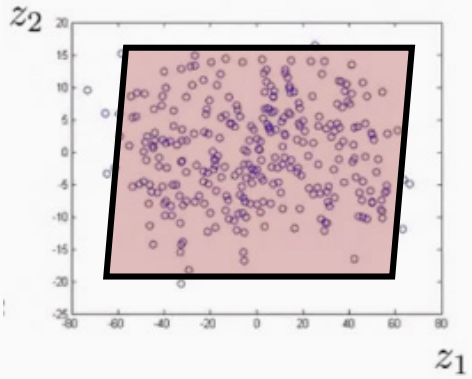
$$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$
$$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$
$$\vdots$$
$$x^{(m)} \in \mathbb{R}^2 \rightarrow z^{(m)} \in \mathbb{R}$$

reducing the data from 3D to 2D:

$x^{(i)} \in \mathbb{R}^3$







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

applying dimensionality reduction to a dataset of m examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$ will result in 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 $k \leq n$.

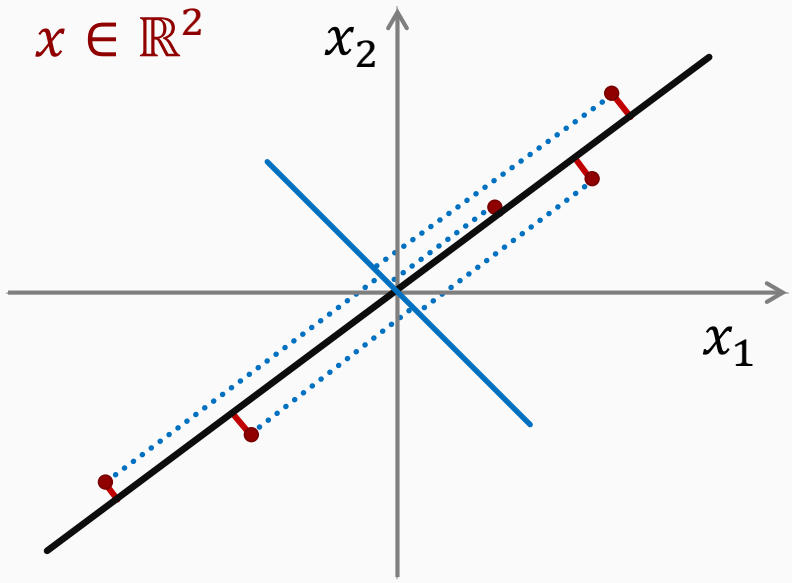
$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$$

principal components analysis: problem formulation

pca finds a lower dimensional surface (a line in many cases) to project the data so that the sum of squares of the distances between the points and the lines is minimized
this is often referred to as the projection error

before applying pca, it is standard practice to perform mean normalization and feature scaling so that the features x_1 and x_2 have zero mean and comparable value ranges

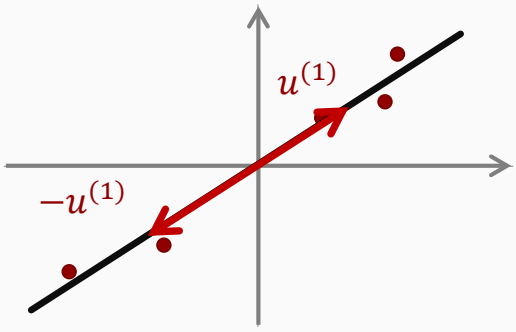
the secondary line in the examples represents a poor choice to project that data onto due to the large distance the that would exist when projecting that data onto the line. principal component analysis will avoid choosing such a line for data projection

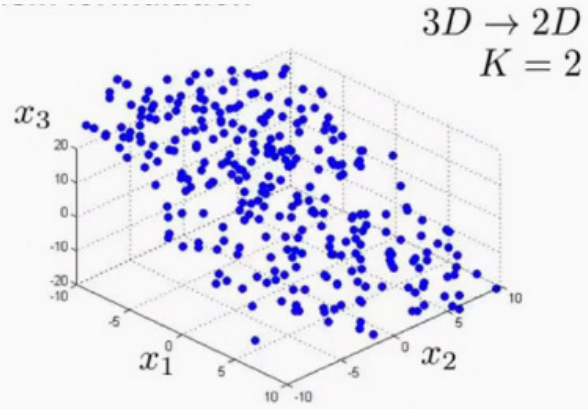


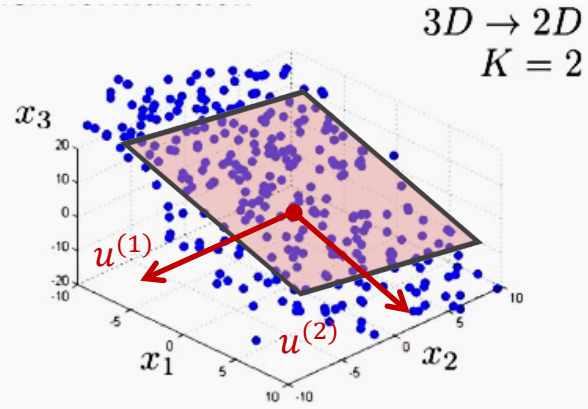
formally stated in the example:
principal component analysis reduces the data from 2-dimensions to 1-dimensions: find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error

It is irrelevant whether $u^{(1)}$ or $-u^{(1)}$ is returned as they both represent the same linear equation onto which to project the data points onto

formally stated in more general terms:
principal component analysis reduces the data from n-dimensions to k-dimensions: find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data so as to minimize the projection error

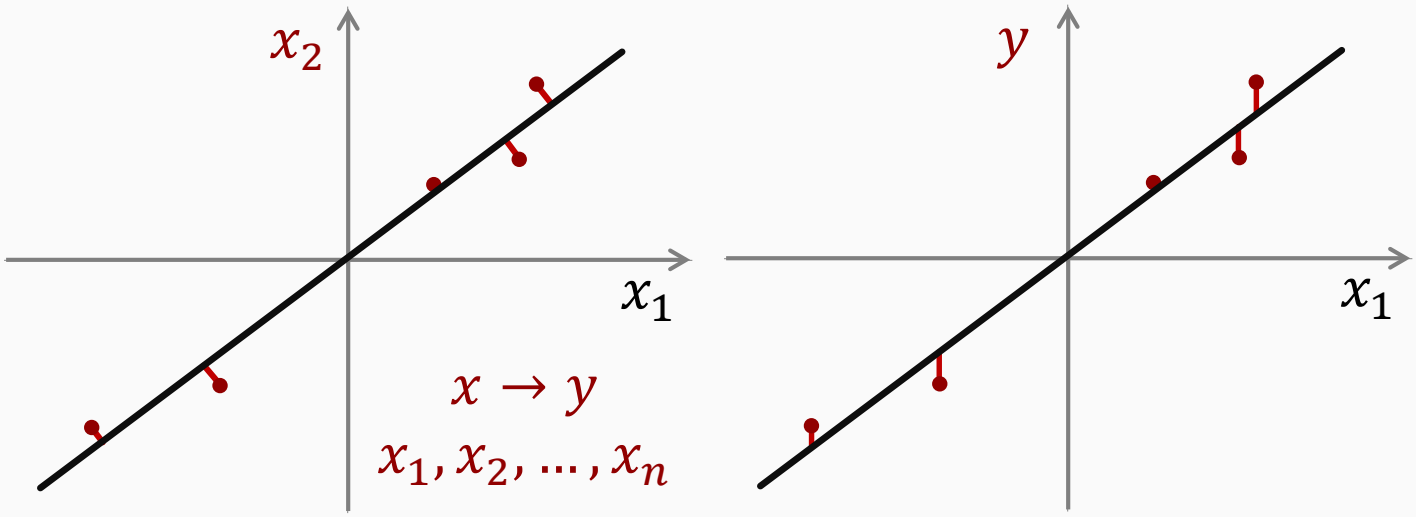


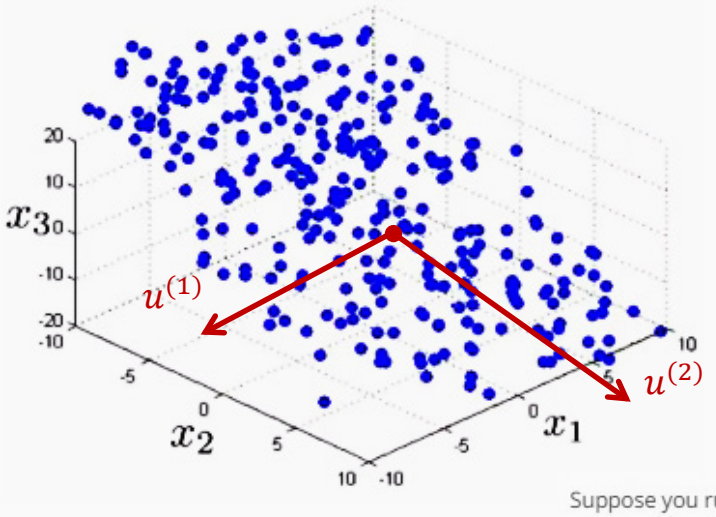




principal component analysis is **not** linear regression

linear regression (left example) projects the data onto a line in reference to its corresponding axis. pca (right example) computes the smallest orthogonal distance between the data points and the line onto which the data is projected

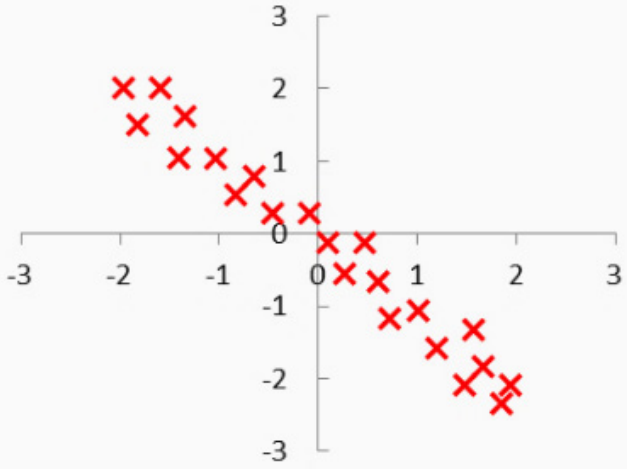




Suppose you run PCA on the dataset below. Which of the following would be a reasonable vector $u^{(1)}$ onto which to project the data? (By convention, we choose $u^{(1)}$ so that $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$, the length of the vector $u^{(1)}$, equals 1.)

- $u^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- $u^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- $u^{(1)} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$
- $u^{(1)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

Correct Response



principal components analysis algorithm

data preprocessing

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

preprocessing (feature scaling/mean normalization):

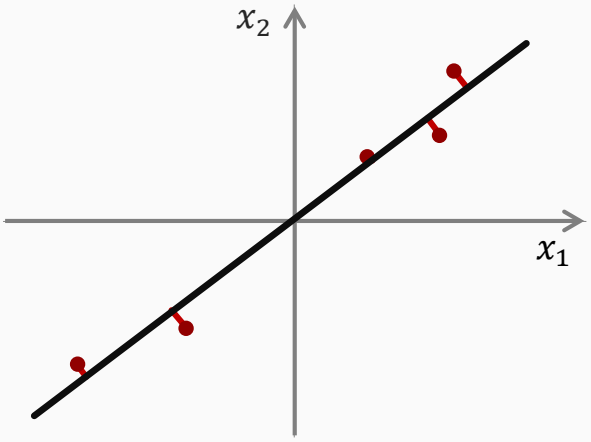
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

compute mean (μ_j) of each feature and then replace each $x_j^{(i)}$ with $x_j - \mu_j$

scale features to have comparable value ranges :

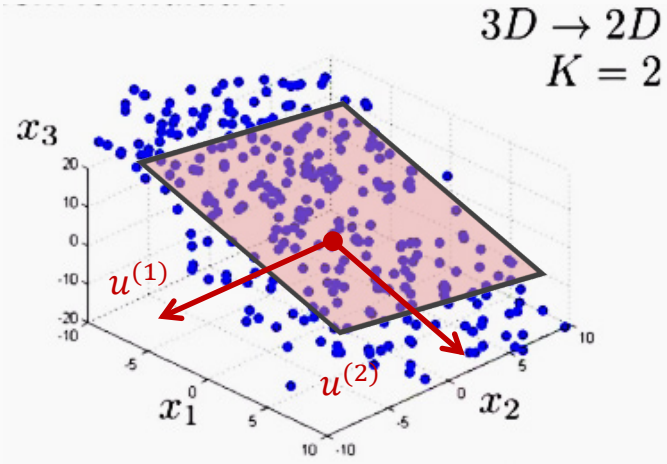
$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

principal component analysis (pca) algorithm



reduce data from 2D to 1D

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



reduce data from 3D to 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

reduce data from n-dimensions to k-dimensions

compute covariance matrix:

$$\sum(\text{sigma}) = \frac{1}{m}\sum_{i=1}^{n}\left(x^{(i)}\right)\left(x^{(i)}\right)^T$$

compute eigenvectors of matrix $\sum(\text{sigma})$:

`[U, S, V] = svd(Sigma);` (single value decomposition, similar to `eig(Sigma)`)

`Sigma` is in n x n matrix and `[U, S, V]` are the matrix outputs

`[U, S, V] = svd(Sigma)` produces:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(m)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

in order to reduce the dimensions of the data from n-dimensions to k-dimensions, take the k vectors $(u^{(1)}, \dots, u^{(k)})$ that will provide the k-direction on which to project the data

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(\boldsymbol{n})} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times \boldsymbol{n}}$$

$$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$$

take the original dataset $x \in \mathbb{R}^n$ and find a lower dimensional representation $z \in \mathbb{R}^n$
this is achieved by taking the first k-columns of the U-matrix: (example of single training example $x^{(i)}$)

$$z^{(i)} = U_{\boldsymbol{reduce}} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(\boldsymbol{k})} \\ | & | & & | \end{bmatrix}^T x^{(i)} \in \mathbb{R}^{n \times \boldsymbol{k}} = \begin{bmatrix} - & \left(u^{(1)}\right)^T & - \\ - & \vdots & - \\ - & \left(u^{(k)}\right)^T & - \end{bmatrix} x^{(i)} \in \mathbb{R}^{\boldsymbol{k} \times 1}$$

after taking the first k-columns of the U-matrix, a reduced version of the original U-matrix's dimensions. z will be computed as $U_{\boldsymbol{reduce}}$ transposed times x. the result are vectors now in rows. the transposed matrix will be a $k \times n$ matrix, x is a $n \times 1$ matrix, so the product of the two is a $k \times 1$ matrix

note: a vectorized implementation of computing the matrix `Sigma` in Octave

if data is provided in the form $\mathrm{X} = \begin{bmatrix} - & \left(x^{(1)}\right)^T & - \\ - & \vdots & - \\ - & \left(x^{(m)}\right)^T & - \end{bmatrix}$ the implantation

`Sigma = (1/m)*X'*X;` creates a vectorized implementation

after Sigma has been computed, compute the matrices:

```
[U, S, V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

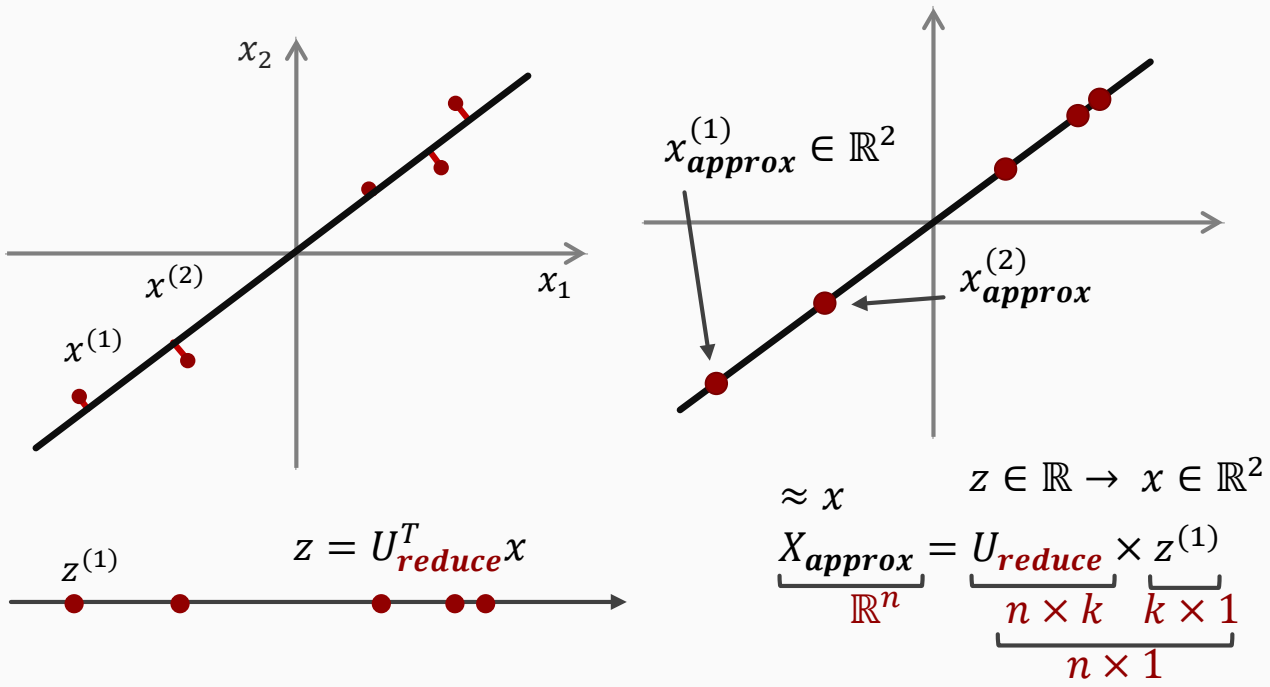
example: in pca, $z \in \mathbb{R}^k$ is obtained from $x \in \mathbb{R}^n$ as follows:

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

the correct expression of z_j is $z_j = \left(u^{(j)}\right)^T x$

applying pca

reconstruction from compressed representation



after pca has been utilized to project the data onto a line by measuring the minimum orthogonal distance, the data will be decompressed as illustrated above

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 is a good thought exercise.) Recall that the percent / fraction of variance retained is given by: $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$ Which of the following will be true? Check all that apply.

- [x] U_{reduce} will be an $n \times n$ matrix.

  Correct Response

- [x] $x_{\text{approx}} = x$ for every example x.

  Correct Response

- [x] The percentage of variance retained will be 100%.

  Correct Response

- [ ] We have that $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} > 1$.

  Correct Response

choosing the number of principal components

attempt to minimize the average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2$

total variation in the data measured as average distance from the origin: $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2$

typically, k will be chosen to be the smallest value so that the average squared projection error (average distance between x and its projections) divided by the total variation of the data (how much the data varies. this value is ideally ≤ 0.01 (1%):

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2} \leq 0.01 \quad (1\%) \quad (5\%, 10\%, \dots)$$

k is chosen so that 99% of variance is retained (sometimes 95%, 90%, 85% ...)

algorithm:

apply pca with $k = 1$

compute $U_{reduce}, z^{(1)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$

check if $\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{approx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2} \leq 0.01$

if $\not\leq 0.01$ with $k = 1$, attempt $k = 2, k = 3, \dots$

```
[U, S, V] = svd(Sigma)
```

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

pick smallest value of k for which

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

(99% of variance retained in dataset)

Previously, we said that PCA chooses a direction $u^{(1)}$ (or k directions $u^{(1)}, \dots, u^{(k)}$) 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:

- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$
- $\frac{1}{m}\sum_{i=1}^{m}\|x_{\text{approx}}^{(i)}\|^2$
- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-x_{\text{approx}}^{(i)}\|^2$

Correct Response

- $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}+x_{\text{approx}}^{(i)}\|^2$

application of principal component analysis

pca can be used to increase the running time speed of an algorithm

supervised learning speedup:

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$ $\quad\quad$ $x^{(i)} \in \mathbb{R}^{10,000}$

extract inputs:

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

↓ apply principal component analysis

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

new training set:

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

the training examples will now be represented with z and will be applied forward

new x examples will be mapped through the same process and applied forward

note: mapping $x \rightarrow z$ should be defined by running pca only on the training set. this mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets. (e.g. finding U_{reduce} should only be mapped on the training set)

summary of principal component analysis application

compression

- reduce memory/disk needed to store data
- speed up learning algorithm
  - choose k by % of variance retained

visualization

- $k = 2$ or $k = 3$

poor application of principal component analysis

using $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$ with the intention of assuming fewer features will make an algorithm less likely to overfit

regularization should be used instead to reduce overfitting:

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

principal component analysis should not be used when:

design of the machine learning system:

get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

✗ run pca to reduce $x^{(i)}$ in dimension to get $z^{(i)}$

train logistic regression on $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})\}$

test on test set: map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. run $h_\theta(z)$ on $\left\{\left(z_{test}^{(1)}, z_{test}^{(1)}\right), \dots, \left(z_{test}^{(m)}, z_{test}^{(m)}\right)\right\}$

instead, apply the machine learning process entirely without pca

before impementing pca, attempt running the original/raw data $x^{(i)}$. only if that does not perform as expected, then implement pca and consider using $z^{(i)}$

Which of the following figures correspond to possible values that PCA may return for $u^{(1)}$ (the first eigenvector / first principal component)? Check all that apply (you may have to check more than one figure).