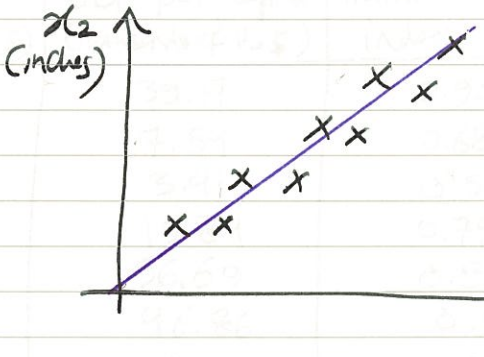
DIMENSIONALITY REDUCTION

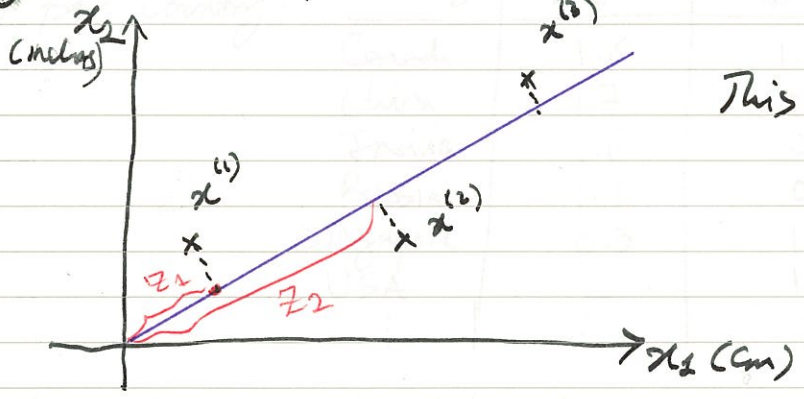
Lecture 82: Motivation I: Data Compression

Suppose we have a data set with 100s of features. Unknown to us, x_1 is the length of something in cm & x_2 is the length in inches. This is a redundant representation:



If x_1 is rounded to the nearest cm & x_2 is rounded to the nearest inch, there could be some noise in the conversion.

In this case, could we reduce the dimension from 2D to 1D? We could project the data points onto the drawn line:



This way: $x^{(1)} \in \mathbb{R}^2 \Rightarrow z^{(1)} \in \mathbb{R}$

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

$\vdots$

Dimensionality reduction can help us with disk space, memory, and more importantly speed up our algorithms. In a real world example, we may want to reduce a 1000D data set to 100D.

Lecture 83: Motivation II: Data Visualization

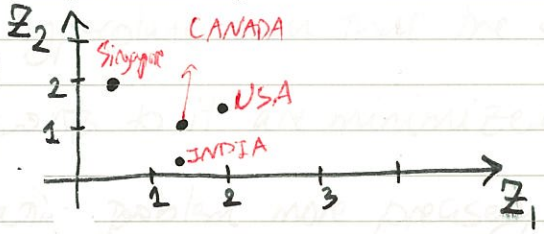
Dimensionality reduction can also be used for visualizing high-dimensional data, by reducing the dimension to 1, 2, or 3 so we can plot the reduced dataset. For instance, let's say we've collected data about many countries:

Country	x_1 GDP (trillions of US$)	x_2 GDP per Capita (thousands of US$)	x_3 Human-development index	x_4 Life expentency	...
Canada	1.577	39.17	0.908	80.7	
China	5.878	7.54	0.687	73	
India	1.632	3.41	0.547	64.7	
Russia	1.48	19.84	0.755	65.5	
Singapore	0.223	56.69	0.866	80	
USA	14.527	46.86	0.91	78.3	
⋮	⋮	⋮	⋮		

We can use dimensionality reduction to compress the features down to two per country:

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
⋮	⋮	⋮

We can then plot this 2D data set:



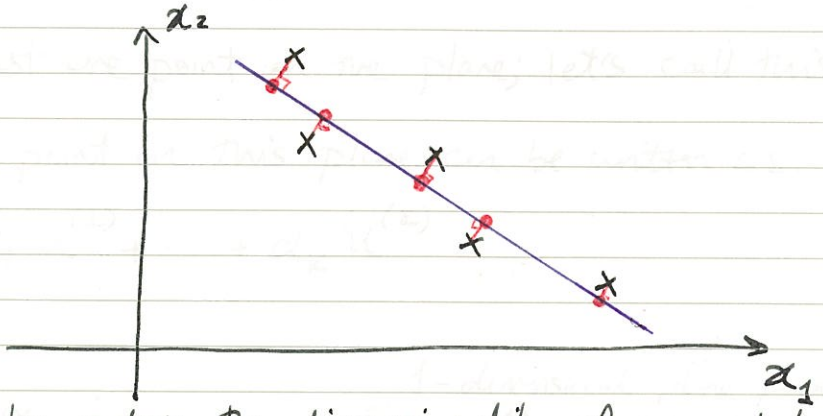
It may not be obvious what z_1 & z_2 mean at first sight, but maybe by looking at the graph:

$z_1 \sim$ Overall country size

$z_2 \sim$ Per person GDP / economic activity.

Lecture 84: Principal Component Analysis Problem Formulation

The most popular algorithm used for dimensionality reduction is Principal Component Analysis (PCA). Consider a two-dimensional data set $(x^{(i)} \in \mathbb{R}^2)$ as follows:

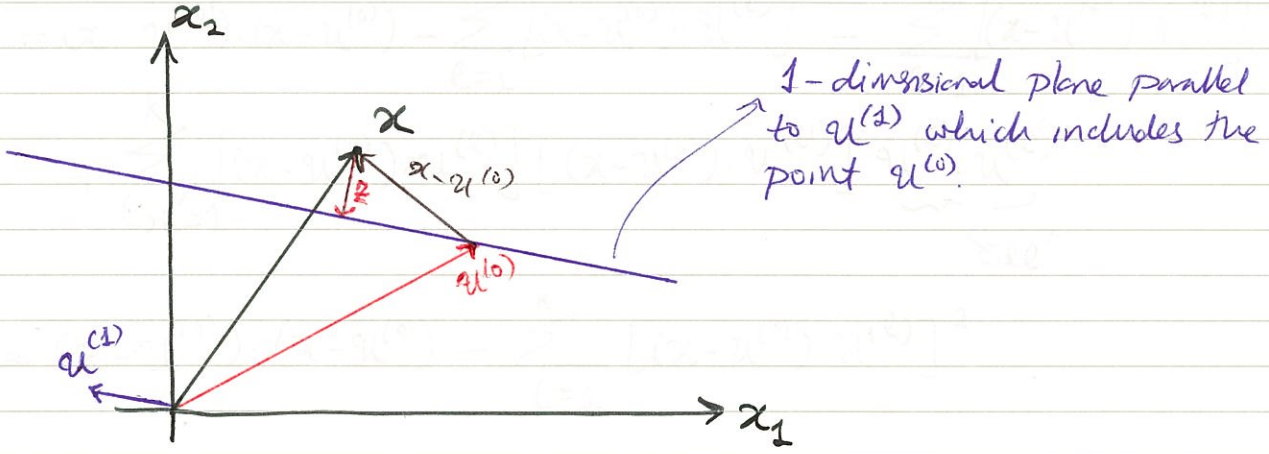


Let's say we want to reduce the dimensionality of our data to 1. It looks like projecting the data onto the blue line above is reasonable. Why is that? Because the projection errors (drawn in red) are small. Therefore, if we want a line to project the data onto, it makes sense to find one with the least overall projection error. This is exactly what PCA does: it minimizes the squared distance of points, their sum that is, to find the optimal line for projection.

In the more general case where the data set is n-dimensional, i.e $x^{(i)} \in \mathbb{R}^n$, and we want to reduce the dimension to $k < n$, PCA finds a k-dimensional hyperplane such that the sum of squared distances of the data points to it are minimized. Let's formulate the optimization problem more precisely.

How can we parametrize a k-dimensional hyperplane? We'll look for k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)} \in \mathbb{R}^n$ which form an orthonormal basis of the plane: $u^{(l)} \cdot u^{(l')} = \delta_{ll'}$ $l, l' = 1, \dots, k$. We also need to know at least one point on the plane; let's call this $u^{(0)} \in \mathbb{R}^n$. Then, any point on this plane can be written as:

$$r = u^{(0)} + \alpha_1 u^{(1)} + \dots + \alpha_k u^{(k)}$$



What is the distance of a point $x \in \mathbb{R}^n$ to our k-dimensional plane? We can decompose x into two parts:

$$x - u^{(0)} = z + \alpha_1 u^{(1)} + \dots + \alpha_k u^{(k)}$$

where $z \cdot u^{(l)} = 0 \quad \forall \; l = 1, \dots, k$

$\alpha_1 u^{(1)} + \dots + \alpha_k u^{(k)}$ → projection of $x - u^{(0)}$ onto the plane.

z is in the direction of the normal vector to the plane. Let's find $\alpha_1, \dots, \alpha_k$: $(x - u^{(0)}) \cdot u^{(l)} = \alpha_l$

As a result: $z = (x - u^{(0)}) - \sum_{l=1}^{K} (x - u^{(0)}) \cdot u^{(l)} \, u^{(l)}$

The norm of z, $\|z\|$, is the distance of point x to the the hyperplane. Let's compute the squared distance:

$$\|z\|^2 = z \cdot z$$

$$= \left[(x - u^{(0)}) - \sum_{l=1}^{K} (x - u^{(0)}) \cdot u^{(l)} \, u^{(l)}\right] \cdot \left[(x - u^{(0)}) - \sum_{l=1}^{K} (x - u^{(0)}) \cdot u^{(l)} u^{(l)}\right]$$

$$= (x - u^{(0)}) \cdot (x - u^{(0)}) - \sum_{l=1}^{K} \left[(x - u^{(0)}) \cdot u^{(l)}\right]^2 - \sum_{l=1}^{K} \left[(x - u^{(0)}) \cdot u^{(l)}\right]^2$$

$$+ \sum_{l,l'=1}^{K} \left[(x - u^{(0)}) \cdot u^{(l)}\right] \left[(x - u^{(0)}) \cdot u^{(l')}\right] \underbrace{u^{(l)} \cdot u^{(l')}}_{\delta_{ll'}}$$

$$= (x - u^{(0)}) \cdot (x - u^{(0)}) - \sum_{l=1}^{K} \left[(x - u^{(0)}) \cdot u^{(l)}\right]^2$$

Our minimization problem, then, is:

$$\min_{u} J(u) = \sum_{i=1}^{m} (x^{(i)} - u^{(0)}) \cdot (x^{(i)} - u^{(0)}) - \sum_{i=1}^{m} \sum_{l=1}^{K} \left[(x^{(i)} - u^{(0)}) \cdot u^{(l)}\right]^2$$

$$\text{s.t.} \quad u^{(l)} \cdot u^{(l')} = \delta_{ll'} \quad l, l' = 1, \ldots, K$$

We don't expect the solution to the minimization to be unique. For instance, $u^{(0)}$ can be any point on the hyperplane. Also, there are infinitely many orthonormal bases which would span the same plane.

Let's verify these statements.

Claim I: If $u^{(0)}$ is a solution to the minimization problem, so is $\tilde{u}^{(0)} = u^{(0)} + \sum_{l=1}^{K} \beta_l u^{(l)} \quad \forall \beta_l \in \mathbb{R}$. ($\tilde{u}^{(0)}$ is another point on the plane)

Proof. It's enough to show $\|z\|$ for a given example remains unchanged.

$$(x-\tilde{u}^{(0)})\cdot(x-\tilde{u}^{(0)}) = \left[(x-u^{(0)}) - \sum_{l=1}^{K}\beta_l u^{(l)}\right]\cdot\left[(x-u^{(0)}) - \sum_{l=1}^{K}\beta_l u^{(l)}\right]$$

$$= (x-u^{(0)})\cdot(x-u^{(0)}) - 2\sum_{l=1}^{K}\beta_l u^{(l)}\cdot(x-u^{(0)}) + \sum_{l=1}^{K}\beta_l^2$$

$$\sum_{l=1}^{K}\left[(x-\tilde{u}^{(0)})\cdot u^{(l)}\right]^2 = \sum_{l=1}^{K}\left\{\left[(x-u^{(0)}) - \sum_{l'=1}^{K}\beta_{l'} u^{(l')}\right]\cdot u^{(l)}\right\}^2$$

$$= \sum_{l=1}^{K}\left[(x-u^{(0)})\cdot u^{(l)} - \beta_l\right]^2$$

$$= \sum_{l=1}^{K}\left\{\left[(x-u^{(0)})\cdot u^{(l)}\right]^2 - 2\beta_l(x-u^{(0)})\cdot u^{(l)} + \beta_l^2\right\}$$

$$(x-\tilde{u}^{(0)})\cdot(x-\tilde{u}^{(0)}) - \sum_{l=1}^{K}\left[(x-\tilde{u}^{(0)})\cdot u^{(l)}\right]^2$$

$$= (x-u^{(0)})\cdot(x-u^{(0)}) - \sum_{l=1}^{K}\left[(x-u^{(0)})\cdot u^{(l)}\right]^2 = \|z\|^2 \quad \checkmark$$

Claim II: If $\{u^{(l)}\}_{l=1\text{-}K}$ are solutions to the minimization problem, so are $\tilde{u}^{(l)} = \sum_{l'=1}^{K}\alpha_{ll'}u^{(l)}$ where $\alpha^T\alpha = \alpha\alpha^T = \mathbb{1}_K$.

($\tilde{u}^{(l)}$ are another orthonormal basis of the plane spanned by $u^{(l)}$s.)

Proof. It's enough to show $\sum_{l=1}^{K}\left[(x-u^{(0)})\cdot\tilde{u}^{(l)}\right]^2 = \sum_{l=1}^{K}\left[(x-u^{(0)})\cdot u^{(l)}\right]^2$.

$$\sum_{l=1}^{K}\left[(x-u^{(0)})\cdot\tilde{u}^{(l)}\right]^2 = \sum_{l=1}^{K}\left[\sum_{l'=1}^{K}\alpha_{ll'}u^{(l)}\cdot(x-u^{(0)})\right]^2$$

$$\sum_{\ell=1}^{K}\left[(x-u^{(0)})\cdot u^{(\ell)}\right]^2 = \sum_{\ell=1}^{K}\sum_{\ell',\ell''=1}^{K} \alpha_{\ell\ell'}\alpha_{\ell\ell''}\,(x-u^{(0)})\cdot u^{(\ell')}\,(x-u^{(0)})\cdot u^{(\ell'')}$$

$$= \sum_{\ell',\ell''=1}^{K}\Big[\underbrace{\sum_{\ell=1}^{K}\alpha_{\ell\ell'}\alpha_{\ell\ell''}}_{\delta_{\ell'\ell''}}\Big](x-u^{(0)})\cdot u^{(\ell')}\,(x-u^{(0)})\cdot u^{(\ell'')}$$

$$= \sum_{\ell'=1}^{K}\left[(x-u^{(0)})\cdot u^{(\ell')}\right]^2 \qquad \checkmark$$

This is not surprising because the quantity whose invariance we just proved is the squared norm of the projection of $x-u^{(0)}$ onto the plane, which should be independent of the orthonormal basis choice.

Suppose we have solved the minimization problem. How do we project $x^{(i)}\in\mathbb{R}^n$ onto a K-dimensional vector? Well, projection of $x^{(i)}-u^{(0)}$ onto the plane is given by

$$\sum_{\ell=1}^{K}\underbrace{(x^{(i)}-u^{(0)})\cdot u^{(\ell)}}\,u^{(\ell)}$$

Coordinates of the projection of $x^{(i)}-u^{(0)}$ in the basis $\{u^{(\ell)}\}$.

Therefore, we can do the mapping as follows:

$$\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}_{n\times 1} \longrightarrow \begin{bmatrix} (x^{(i)}-u^{(0)})\cdot u^{(1)} \\ \vdots \\ (x^{(i)}-u^{(0)})\cdot u^{(K)} \end{bmatrix}_{K\times 1} \equiv p^{(i)}$$

What we use $\tilde{u}^{(0)}$ & $\{\tilde{u}^{(\ell)}\}$ instead of $u^{(0)}$ & $\{u^{(\ell)}\}$? Then the resulting $\tilde{p}$ would be related to p by a rotation & translation:

$$\tilde{p}_\ell = (x-\tilde{u}^{(0)})\cdot\tilde{u}^{(\ell)} = \Big[(x-u^{(0)})-\sum_{\ell=1}^{K}\beta_\ell u^{(\ell)}\Big]\cdot\Big[\sum_{\ell=1}^{K}\alpha_{\ell\ell'}u^{(\ell')}\Big] = \big[\alpha(p-\beta)\big]_\ell$$

(rotation: α; translation: β)

Lecture 85: Principal Component Analysis Algorithm

Let's tackle the optimization problem on page 27.

$$\frac{\partial J}{\partial u_j^{(0)}} = -2\sum_{i=1}^{m}(x_j^{(i)} - u_j^{(0)}) + 2\sum_{i=1}^{m}\sum_{l=1}^{k}(x^{(i)} - u^{(0)})\cdot u^{(l)} u_j^{(l)} = 0$$

$$\Rightarrow -\sum_{i=1}^{m} x_j^{(i)} + m u_j^{(0)} + \sum_{l=1}^{k}\left[\sum_{i=1}^{m} x^{(i)} - m u^{(0)}\right]\cdot u^{(l)} u_j^{(l)} = 0$$

Let $\bar{x} \equiv \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ be the mean vector across all examples.

Then the above equation is satisfied if $\boxed{u^{(0)} = \bar{x}}$, which means the mean of all data points must be on the optimal hyperplane.

Let $\hat{x}^{(i)} \equiv x^{(i)} - \bar{x}$, then:

$$J(u) = \sum_{i=1}^{m} \hat{x}^{(i)}\cdot\hat{x}^{(i)} - \sum_{i=1}^{m}\sum_{l=1}^{k}\left[\hat{x}^{(i)}\cdot u^{(l)}\right]^2$$

Since the first term is independent of $u^{(l)}$, minimizing $J(u)$ is equivalent to maximizing the second term:

$$\max_{u} \tilde{J}(u) = \sum_{i=1}^{m}\sum_{l=1}^{k}\left[\hat{x}^{(i)}\cdot u^{(l)}\right]^2$$

$$\text{s.t.} \quad u^{(l)}\cdot u^{(l')} = \delta_{ll'} \quad l, l' = 1, \ldots, k$$

We can rewrite this in a more concise way by defining the following matrices:

$$\boxed{U_{jl} \equiv \left[u^{(l)}\right]_j}$$ $n \times k$ matrix

$$\boxed{\hat{X}_{ij} \equiv \hat{x}_j^{(i)}}$$ $m \times n$ mean-zero design matrix

$$\tilde{J}(u) = \sum_{i=1}^{m} \sum_{l=1}^{k} \left[\sum_{j=1}^{n} \hat{x}_j^{(i)} u_j^{(l)} \right] \left[\sum_{j=1}^{n} \hat{x}_j^{(i)} u_j^{(l)} \right]$$

$$= \sum_{i=1}^{m} \sum_{l=1}^{k} [\hat{X} U]_{il} [\hat{X} U]_{il}$$

$$= \sum_{l=1}^{k} \sum_{i=1}^{m} [U^T \hat{X}^T]_{li} [\hat{X} U]_{il}$$

$$= \sum_{l=1}^{k} [U^T \hat{X}^T \hat{X} U]_{ll} = tr(U^T \hat{X}^T \hat{X} U)$$

Note that: $U = \left[\left[u^{(1)} \right] \left[u^{(2)} \right] \cdots \left[u^{(k)} \right] \right]$

so $u^{(l)} \cdot u^{(l')} = \delta_{ll'}$ is equivalent to $U^T U = \mathbb{I}_k$ because

$$U^T U = \begin{bmatrix} [& u^{(1)T} &] \\ [& \vdots &] \\ [& u^{(k)T} &] \end{bmatrix} \left[\left[u^{(1)} \right] \left[u^{(2)} \right] \cdots \left[u^{(k)} \right] \right]$$

$$= \begin{bmatrix} u^{(1)} \cdot u^{(1)} & u^{(1)} \cdot u^{(2)} & \cdots & u^{(1)} \cdot u^{(k)} \\ u^{(2)} \cdot u^{(1)} & u^{(2)} \cdot u^{(2)} & \cdots & u^{(2)} \cdot u^{(k)} \\ \vdots & & & \\ u^{(k)} \cdot u^{(1)} & u^{(k)} \cdot u^{(2)} & \cdots & u^{(k)} \cdot u^{(k)} \end{bmatrix} = \mathbb{I}_k$$

Let $\Sigma \equiv \hat{X}^T \hat{X}$, which is an $n \times n$ matrix. Our optimization problem is then:

$$\max_U \tilde{J}(U) = tr(U^T \Sigma U)$$

$$\text{s.t.} \quad U^T U = \mathbb{I}_k$$

Note that Σ is symmetric positive semi-definite:

* $\Sigma^T = (\hat{X}^T\hat{X})^T = \hat{X}^T\hat{X} = \Sigma$
* $z^T\Sigma z = z^T\hat{X}^T\hat{X}z = (\hat{X}z)^T(\hat{X}z) \geq 0 \quad \forall\ z \in \mathbb{R}^n$

As a result, Σ is diagonalizable & all of its eigen values are positive:

$$\Sigma v^{(j)} = \lambda_j v^{(j)} \quad \text{s.t. } v^{(j)} \cdot v^{(j')} = \delta_{jj'} \ \&\ \lambda_j \geq 0.$$

Let $V_{ij} \equiv v_i^{(j)}$ be the matrix of all eigen vectors:

$$V = \left[\left[v^{(1)}\right]\left[v^{(2)}\right]\cdots\left[v^{(n)}\right]\right]$$

Then: $\Sigma = VDV^T$ where $D_{ij} = \lambda_i \delta_{ij}$.

In what follows, we'll assume $\lambda_1 > \lambda_2 > \cdots > \lambda_n$. In other words, $v^{(1)}$ is the eigen vector corresponding to the largest eigenvalue, $v^{(2)}$ is the eigenvector corresponding to the second-largest eigenvalue, and so on. We then have:

$$U^T\Sigma U = U^TVDV^TU = (V^TU)^TDV^TU$$

Let $\boxed{B \equiv V^TU}$ ($n \times k$ matrix)

Then $\tilde{J} = \mathrm{tr}(B^TDB)$. Also: $B^TB = U^TVV^TU = U^TU = \mathbb{I}_k$

Therefore, we can rewrite our optimization problem as follows:

$$\max_B \tilde{J}(B) = \mathrm{tr}(B^TDB) \quad \text{s.t. } B^TB = \mathbb{I}_k$$

$$\tilde{J} = \mathrm{tr}(B^T D B) = \sum_{k=1}^{K} [B^T D B]_{\ell\ell} = \sum_{\ell=1}^{K} \sum_{j,j'=1}^{n} [B^T]_{\ell j} D_{jj'} B_{j'\ell}$$

$$= \sum_{\ell=1}^{K} \sum_{j=1}^{n} \lambda_j B_{j\ell}^2$$

$$\Rightarrow \boxed{\tilde{J} = \sum_{j=1}^{n} \lambda_j \beta_j} \quad \text{where} \quad \boxed{\beta_j \equiv \sum_{k=1}^{K} B_{j\ell}^2}$$

Note that $\sum_{j=1}^{n} \beta_j = \sum_{\ell=1}^{K} \sum_{j=1}^{n} B_{j\ell}^2 = \sum_{\ell=1}^{K} [B^T B]_{\ell\ell} = \sum_{\ell=1}^{K} [\mathbb{1}_K]_{\ell\ell}$

$= K.$

So: $\boxed{\sum_{j=1}^{n} \beta_j = K}$ ⊕

Claim I: $\tilde{J} \leq \sum_{j=1}^{K} \lambda_k$, i.e. the maximum value $\tilde{J}$ can take is the sum of the K^{th} largest eigenvalues of Σ.

Proof. First we will show that $\beta_j \leq 1$. Let's rewrite $B_{j\ell}$:

$$B_{j\ell} = [V^T U]_{j\ell} = \sum_{j'=1}^{n} V_{jj'} U_{j'\ell} = \sum_{j'=1}^{n} v_{j'}^{(j)} u_{j'}^{(\ell)} = v^{(j)} \cdot u^{(\ell)}$$

Then: $\beta_j = \sum_{\ell=1}^{K} (v^{(j)} \cdot u^{(\ell)})^2$.

Consider now $z^{(j)} \equiv v^{(j)} - \sum_{\ell=1}^{K} v^{(j)} \cdot u^{(\ell)} u^{(\ell)}$

$$\|z^{(j)}\|^2 = \|v^{(j)}\|^2 - \sum_{\ell=1}^{K} [v^{(j)} \cdot u^{(\ell)}]^2 = 1 - \beta_j$$

Since $\|z^{(j)}\|^2 \geq 0 \Rightarrow \boxed{\beta_j \leq 1}$. Of course, $\beta_j \geq 0$.

From ⊕: $\beta_1 + \beta_2 + \cdots + \beta_n = K$. Also $\tilde{J} = \lambda_1 \beta_1 + \lambda_2 \beta_2 + \cdots + \lambda_n \beta_n$.

Remember that we're assuming $\lambda_1 > \lambda_2 > \cdots > \lambda_n$.

$$\tilde{J} = \lambda_1\beta_1 + \cdots + \lambda_k\beta_k + \lambda_{k+1}\beta_{k+1} + \cdots + \lambda_n\beta_n$$

$$\leq \lambda_1\beta_1 + \cdots + \lambda_k\beta_k + (\beta_{k+1} + \cdots + \beta_n)\lambda_{k+1}$$

↳ since $\lambda_{k+1} > \lambda_{k+2}, \cdots, \lambda_n$ & $\beta_{k+1}, \cdots, \beta_n \geq 0$.

$$\Rightarrow \tilde{J} \leq [\lambda_1 + \cdots + \lambda_k] + [(\beta_1 - 1)\lambda_1 + \cdots + (\beta_k - 1)\lambda_k]$$

$$+ [(1-\beta_1) + (1-\beta_2) + \cdots + (1-\beta_k)]\lambda_{k+1}$$

↳ here we're using $\beta_1 + \cdots + \beta_k + \beta_{k+1} + \cdots + \beta_n = k$

$$\Rightarrow \beta_{k+1} + \cdots + \beta_n = k - \beta_1 - \cdots + \beta_k$$

$$= (1-\beta_1) + \cdots + (1-\beta_k)$$

Finally $\tilde{J} \leq \lambda_1 + \cdots + \lambda_k$

$$+ \underbrace{(\beta_1 - 1)}_{\leq 0}\underbrace{(\lambda_1 - \lambda_{k+1})}_{\geq 0} + \cdots + \underbrace{(\beta_k - 1)}_{\leq 0}\underbrace{(\lambda_k - \lambda_{k+1})}_{\geq 0}$$

$$\leq \lambda_1 + \cdots + \lambda_k. \quad \checkmark$$

Claim II: $\tilde{J}$ obtains its maximum value $\sum_{l=1}^{k} \lambda_k$ when $u^{(l)} = v^{(l)}$ $l = 1, \cdots, k$

Proof. We had already shown that $\beta_j = \sum_{l=1}^{k} (v^{(j)} \cdot u^{(l)})^2$.

If we let $u^{(l)} = v^{(l)}$ $l = 1, \cdots, k$, we have $\beta_1 = \beta_2 = \cdots = \beta_k = 1$

& $\beta_{k+1} = \beta_{k+2} = \cdots = \beta_n = 0$, from which it follows that

$\tilde{J} = \lambda_1\beta_1 + \cdots + \lambda_n\beta_n = \lambda_1 + \cdots + \lambda_k$. ✓

It follows from claims I & II that the solution to our optimization problem is that our k-dimensional plane should be spanned by the k eigenvectors

of Σ with the highest eigenvalues. That's just simple and beautiful.

$$U = \begin{bmatrix} \begin{bmatrix} & & \\ v^{(1)} & & \end{bmatrix} \cdots \begin{bmatrix} & & \\ v^{(k)} & & \end{bmatrix} \end{bmatrix}$$

On page 29 we argued that mapping $x^{(i)}$ to a k-dimensional vector can be done as follows:

$$\begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \rightarrow \begin{bmatrix} \hat{x}^{(i)} \cdot v^{(1)} \\ \vdots \\ \hat{x}^{(i)} \cdot v^{(k)} \end{bmatrix} = \begin{bmatrix} [& v^{(1)T} &] \\ & \vdots & \\ [& v^{(k)T} &] \end{bmatrix} \begin{bmatrix} \hat{x}_1^{(i)} \\ \vdots \\ \hat{x}_n^{(i)} \end{bmatrix} = U^T \hat{x}^{(i)}$$

We can do this for all examples by simply computing $\hat{X}U$

$$\hat{X}U = \begin{bmatrix} [& \hat{x}^{(1)T} &] \\ [& \hat{x}^{(2)T} &] \\ & \vdots & \\ [& \hat{x}^{(m)T} &] \end{bmatrix} \begin{bmatrix} \begin{bmatrix} & & \\ v^{(1)} & & \end{bmatrix} \begin{bmatrix} & & \\ v^{(2)} & & \end{bmatrix} \cdots \begin{bmatrix} & & \\ v^{(k)} & & \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{x}^{(1)} \cdot v^{(1)} & \cdots & \hat{x}^{(1)} \cdot v^{(k)} \\ \hat{x}^{(2)} \cdot v^{(1)} & \cdots & \hat{x}^{(2)} \cdot v^{(k)} \\ \vdots & & \\ \hat{x}^{(m)} \cdot v^{(1)} & \cdots & \hat{x}^{(m)} \cdot v^{(k)} \end{bmatrix}$$

$m \times n$ $\quad$ $n \times k$ $\quad$ $m \times k$

our m examples are now k-dimensional.

PCA Summary

Let $\bar{x} \equiv \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$, $\hat{x}^{(i)} \equiv x^{(i)} - \bar{x}$, $\hat{X}_{ij} \equiv \hat{x}_j^{(i)}$

Diagonalize $\Sigma \equiv \hat{X}^T \hat{X}$ so that $\Sigma = V D V^T$ where

$V^T V = V V^T = 1_n$ & $D = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & \lambda_n \end{bmatrix}$ s.t. $\lambda_1 > \lambda_2 > \cdots \lambda_n$.

Let $\underset{\sim}{U}$ denote the first k columns of V. Map examples to k-dimensions:

$n \times k$ matrix

$$X \rightarrow \hat{X}\underset{\sim}{U} \quad \text{or} \quad x^{(i)} \rightarrow \underset{\sim}{U}^T x^{(i)}$$

PCA has a nice probabilistic interpretation as well. Let's look more closely at Σ:

$$\Sigma_{jj'} = [\hat{X}^T\hat{X}]_{jj'} = \sum_{i=1}^{m} [\hat{X}^T]_{ji}\hat{X}_{ij'} = \sum_{i=1}^{m} \hat{X}_{ij}\hat{X}_{ij'} = \sum_{i=1}^{m} \hat{x}_j^{(i)}\hat{x}_{j'}^{(i)}$$

$$= \sum_{i=1}^{m} (x_j^{(i)} - \bar{x}_j)(x_{j'}^{(i)} - \bar{x}_{j'})$$

This is the sample covariance matrix of the data set, up to a normalization. Note that multiplying Σ by a positive constant does not affect our optimization results at all, since all eigenvalues of Σ will be scaled by the same number & the eigenvectors will be the same: So let's redefine Σ:

$$\Sigma_{jj'} \equiv \frac{1}{m-1}\sum_{i=1}^{m} (x_j^{(i)} - \bar{x}_j)(x_{j'}^{(i)} - \bar{x}_{j'}) = \frac{1}{m-1}\hat{X}^T\hat{X}$$

So we can think of $x_1, x_2, \ldots, x_n$ as random variables with mean $\bar{x}_1, \ldots, \bar{x}_n$ & covariance matrix given by Σ. In other words, $\{x_j^{(i)}\}_{i=1\text{-}m}$ are realizations of x_j, and so on. In this light, let's make an interesting claim about the reduced variables.

Claim: The reduced K-dimensional data set has uncorrelated features & the variance of each feature is given by the corresponding eigenvalue of Σ.

proof: Our reduced $m \times k$ design matrix is given by $Z \equiv \hat{X}U$.

Its sample covariance matrix Σ^Z is given by:

$$\Sigma^Z = \frac{1}{m-1} Z^T Z = \frac{1}{m-1} (\hat{X} V)^T (\hat{X} V) = \frac{1}{m-1} V^T \hat{X}^T \hat{X} V = V^T \Sigma V$$

$$= V^T V D V^T V$$

$$V^T V = \begin{bmatrix} [& v^{(1)\top} &] \\ & \vdots & \\ [& v^{(k)\top} &] \\ & \vdots & \\ [& v^{(n)\top} &] \end{bmatrix}_{n \times n} \begin{bmatrix} | & & | \\ v^{(1)} & \cdots & v^{(k)} \\ | & & | \end{bmatrix}_{n \times k} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & \cdots & & 1 \\ 0 & \cdots & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & & 0 \end{bmatrix}$$

$\mathbb{I}_k$

$\to (n-k) \times k$ matrix of zeros.

$$\Rightarrow \Sigma^Z_{\ell\ell'} = \lambda_\ell \delta_{\ell\ell'} \qquad \ell, \ell' = 1, \ldots, k$$

Let's also do a non-matrix proof. We have transformed the random variables $x_1, \ldots, x_n$ to $z_1, \ldots, z_k$ where:

$$z_1 = \hat{x} \cdot v^{(1)} = \sum_{j=1}^{n} \hat{x}_j v_j^{(1)}$$

$$\vdots$$

$$z_k = \hat{x} \cdot v^{(k)} = \sum_{j=1}^{n} \hat{x}_j v_j^{(k)}$$

First note that $\langle z_1 \rangle = \cdots = \langle z_k \rangle = 0$ because $\langle \hat{x}_j \rangle = 0$.

$$\Sigma^Z_{\ell\ell'} = \langle z_\ell z_{\ell'} \rangle = \sum_{j_1, j_2 = 1}^{n} v_{j_1}^{(\ell)} v_{j_2}^{(\ell')} \langle \hat{x}_{j_1} \hat{x}_{j_2} \rangle$$

$$= \sum_{j_1, j_2 = 1}^{n} v_{j_1}^{(\ell)} v_{j_2}^{(\ell)} \Sigma_{j_1 j_2} = v^{(\ell)\top} \Sigma v^{(\ell)} = \lambda_\ell v^{(\ell)\top} v^{(\ell')}$$

$$= \lambda_\ell \delta_{\ell\ell'} \qquad \checkmark$$

We see that PCA reduces our n-features into $k \leq n$ uncorrelated

features. We also showed that the variance of the new features are given by the eigenvalues of Σ. Remember that PCA maximizes $tr(U^T \Sigma U) = tr(\Sigma^2)$. As a result, we can think about PCA as follows: Given n correlated random variables $x_1, \ldots, x_n$, PCA finds $K \leq n$ random variables $Z_1, \ldots, Z_K$ which are uncorrelated and retain as much of variance of $x_1, \ldots, x_n$ as possible. In fact, this could be a starting point for deriving PCA from a probabilistic point of view. Let's do this: Consider n random variables $x_1, \ldots, x_n$ with means $\bar{x}_1, \ldots, \bar{x}_n$ & covariance $\Sigma_{ij} = \langle (x_i - \bar{x}_i)(x_j - \bar{x}_j) \rangle$.

Let's find a linear combination of $x_1, \ldots, x_n$ with the highest variance: $Z_1 \equiv \sum_{j=1}^{n} \alpha_j x_j$. Of course, with no constraint on α_j we can make the variance of Z_1 as large as we like, so let's impose $\sum_{j=1}^{n} \alpha_j^2 = 1$. We'll shortly see the meaning of this constraint.

$$VAR(Z_1) = \langle (Z_1 - \bar{Z}_1)^2 \rangle = \sum_{j,j'=1}^{n} \alpha_j \alpha_{j'} \langle (x_j - \bar{x}_j)(x_{j'} - \bar{x}_{j'}) \rangle$$

$$= \sum_{j,j'=1}^{n} \alpha_j \alpha_{j'} \Sigma_{jj'} = \alpha^T \Sigma \alpha$$

So our optimization problem is:

$$\max_{\alpha} J(\alpha) = \alpha^T \Sigma \alpha$$

$$\text{s.t.} \quad \alpha^T \alpha = 1$$

$$L = \alpha^T \Sigma \alpha - \lambda_1(\alpha^T\alpha - 1)$$

$$\frac{\partial L}{\partial \alpha} = 2\Sigma\alpha - 2\lambda_1\alpha = 0 \Rightarrow \Sigma\alpha = \lambda_1\alpha$$

we see that α should be a normalized eigenvector of Σ. But which one? Let's see which one maximizes J:

$$J(\alpha) = \alpha^T\Sigma\alpha = \lambda_1\alpha^T\alpha = \lambda_1.$$

Therefore, we should take α to be the normalized eigenvector of Σ with the highest eigenvalue. Using our previous notation: $\alpha = \nu^{(1)}$.

So we have that: $Z_1 = x \cdot \nu^{(1)}$ where $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$. Let's also require that $\bar{Z}_1 = 0$ so that $Z_1 = \nu^{(1)} \cdot (x - \bar{x}) = \nu^{(1)} \cdot \hat{x}$ where

$$\hat{x}_i = x_i - \bar{x}_i.$$

Let's now find another linear of combination of $\hat{x}_i$ with the following properties: $Z_2 = \sum_{j=1}^{n} \alpha_j \hat{x}_j$ where

(i) $VAR(Z_2)$ is maximized
(ii) $\langle Z_1 Z_2 \rangle = 0$
i.e Z_2 is uncorrelated with Z_1
(iii) $\alpha^T\alpha = 1$.

Note that by construction $\bar{Z}_2 = 0$. Let's formulate our optimization problem: $VAR(Z_2) = \langle Z_2^2 \rangle = \alpha^T\Sigma\alpha$

$$\langle Z_1 Z_2 \rangle = \alpha^T\Sigma\nu^{(1)} = \lambda_1\alpha^T\nu^{(1)} = 0$$

So we have: $\max_{\alpha} J(\alpha) = \alpha^T\Sigma\alpha$

$$\text{s.t. } \alpha^T\nu^{(1)} = 0 \quad \& \quad \alpha^T\alpha = 1$$

$$L = \alpha^T \Sigma \alpha - \lambda_2(\alpha^T\alpha - 1) - \beta\,\alpha^T \nu^{(1)}$$

$$\frac{\partial L}{\partial \alpha} = 0 \Rightarrow 2\Sigma\alpha - 2\lambda_2\alpha - \beta\nu^{(1)} = 0 \quad (\#)$$

$$\frac{\partial L}{\partial \lambda_2} = 0 \Rightarrow \alpha^T\alpha = 1$$

$$\frac{\partial L}{\partial \beta} = 0 \Rightarrow \alpha^T\nu^{(1)} = 0$$

Apply $\nu^{(1)T}$ to $(\#)$: $2\nu^{(1)T}\Sigma\alpha - 2\lambda_2\nu^{(1)T}\alpha - \beta = 0$

$$\Rightarrow 2\underbrace{\alpha^T\Sigma\nu^{(1)}}_{=\alpha^T(\lambda_1\nu^{(1)})} - \beta = 0 \Rightarrow \beta = 0$$

Since $\beta = 0 \Rightarrow \Sigma\alpha = \lambda_2\alpha$, so α has to a normalized eigenvector of Σ. But which one? Well, the only one that's orthogonal to $\nu^{(1)}$ & has the largest eigenvalue is the eigenvector of Σ corresponding to the second largest eigenvalue, i.e $\nu^{(2)}$. $\Rightarrow Z_2 = \hat{x}\cdot\nu^{(2)}$.

We see that if we continue this K times, each time requiring to maximize the variance & being uncorrelated with the previous ones, we'll get the following answer: $Z_1 = \hat{x}\cdot\nu^{(1)}$, $Z_2 = \hat{x}\cdot\nu^{(2)}, \dots, Z_K \equiv \hat{x}\cdot\nu^{(K)}$.

This was precisely our conclusion from the geometric picture: for an example $x^{(i)}$, the reduced vector is given by:

$$\begin{bmatrix} \hat{x}^{(i)}\cdot\nu^{(1)} \\ \vdots \\ \hat{x}^{(i)}\cdot\nu^{(K)} \end{bmatrix} \begin{matrix} \\ \\ \end{matrix}$$

→ A realization of Z_1

→ A realization of Z_K.

Oh, and $Z_1, \dots, Z_K$ are called the principal components.

Here's another cool fact about the geometric vs. probabilistic picture.

Claim: Sum of the squared distnces of $x^{(i)}$ from the mean $\bar{x}$ is equal to the sum of the eigen values of $\hat{X}^T\hat{X}$, i.e. its trace.

Proof:

$$\sum_{i=1}^{m} (x^{(i)} - \bar{x}) \cdot (x^{(i)} - \bar{x}) = \sum_{i=1}^{m} \hat{x}^{(i)} \cdot \hat{x}^{(i)}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \hat{x}_j^{(i)} \hat{x}_j^{(i)} = \sum_{j=1}^{n} \sum_{i=1}^{m} \hat{X}_{ij} \hat{X}_{ij} = \sum_{j=1}^{n} [\hat{X}^T\hat{X}]_{jj}$$

$$= \text{tr}(\hat{X}^T\hat{X}) = \sum_{j=1}^{n} \lambda_j \quad \text{where } \lambda_j \text{ are the eigenvalues of } \hat{X}^T\hat{X}.$$

One important thing to note is that PCA is sensitive to scaling features.

Consider two features with the following properties:

* $\langle x_1 \rangle = \langle x_2 \rangle = 0$
* $\langle x_1^2 \rangle = \langle x_2^2 \rangle = \langle x_1 x_2 \rangle = 1$

Then: $\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ & it can be verified that:

* $u^{(1)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\lambda_1 = 2$
* $u^{(2)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\lambda_2 = 0$

Consider now $\tilde{x}_1 = \alpha x_1$ & $\tilde{x}_2 = x_2$. In this case $\Sigma = \begin{bmatrix} \alpha^2 & \alpha \\ \alpha & 1 \end{bmatrix}$

and:

* $u^{(1)} = \frac{1}{\sqrt{1+\alpha^2}} \begin{bmatrix} \alpha \\ 1 \end{bmatrix}$, $\lambda_1 = 1 + \alpha^2$
* $u^{(2)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\lambda_2 = 0$

For very large α, $v^{(1)} \sim \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ which is very different from the unscaled eigenvector $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. This is unfortunate. Imagine x_1 represents temperature. PCA would produce different results depending on what unit we choose. (e.g. Fahrenheit vs. Celsius). One way of making PCA less arbitrary is to scale all features such that they have unit variance:

$$x_j^{(i)} \rightarrow \frac{x_j^{(i)} - \bar{x}_j}{\sigma_j} \quad \text{where} \begin{cases} \bar{x}_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \\ \sigma_j = \frac{1}{m-1} \sum_{i=1}^{m} (x_j^{(i)} - \bar{x}_j)^2 \end{cases}$$

Therefore, in practice, we should perform feature-scaling before using PCA on our data set.

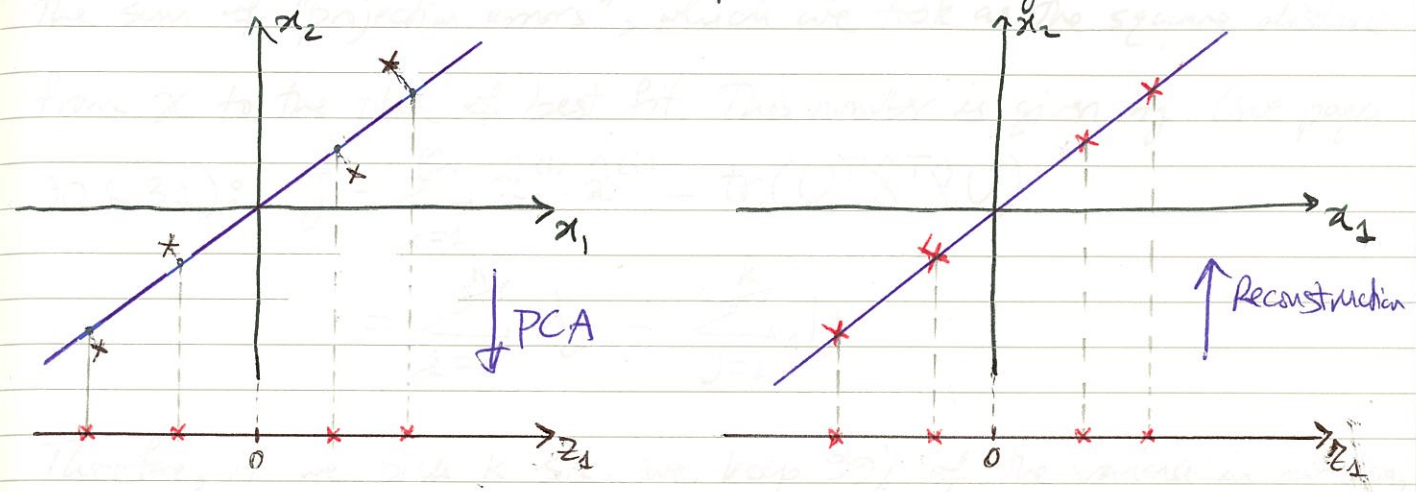
Lecture 86: Reconstruction from Compressed Representation

Given an n-dimensional example x, PCA can reduce its dimension to $k \leq n$ via: $z = U^T x$ (see page 35). Given z, can we reconstruct x? Of course, not entirely. z is simply the projection of x on the hyperplane spanned by k eigenvectors of $X^T X$ with the highest k eigenvalues. (we are assuming mean normalization is applied to our data before it's fed to PCA.). More precisely, $z_1, \ldots, z_k$ are the coordinates of the projection of x corresponding to the basis $v^{(1)}, \ldots, v^{(k)}$. The projected vector is: $\sum_{l=1}^{k} z_l v^{(l)}$ ($z_l \in \mathbb{R}$, $v^{(l)} \in \mathbb{R}^n$)

Let's call this projection x^{approx}.

$$x_j^{approx} = \sum_{l=1}^{k} z_l v_j^{(l)} = \sum_{l=1}^{k} z_l u_j^{(l)} = \sum_{l=1}^{k} z_l U_{jl} = [Uz]_j$$

This is consistent with what we expect from writing $z = U^T x$.



Lecture 87: Choosing the Number of Principal Components

How many principal components should we choose? What's often discussed is how much variance is retained after dimensional reduction. We proved that $z_1, \dots, z_k$ are uncorrelated & their variance is given by $\lambda_1, \dots, \lambda_k$, k highest eigenvalues of $\hat{X}^T\hat{X}$. When $k=n$, we will have captured all the variance in our data set. So it makes sense to pick k s.t.

$$\frac{\sum_{l=1}^{k} \lambda_l}{\sum_{l=1}^{n} \lambda_l} \geq \text{threshold (e.g. 99\%)}.$$

This ratio also has a geometric meaning. Earlier we showed that

the sum of the squared distances of $x^{(i)}$'s from the mean is equal to the sum of the eigen values of $\hat{X}^T\hat{X}$, which we're interpreting as the total variance in our data set. Remember that PCA minimizes the sum of "projection errors", which we took as the square distance from x to the plane of best fit. This number is given by (see pages 30 & 31): $J = \sum_{i=1}^{m} \hat{x}^{(i)} \cdot \hat{x}^{(i)} - tr(U^T\hat{X}^T\hat{X}U)$

$$= \sum_{i=1}^{n} \lambda_j - \sum_{j=1}^{k} \lambda_j$$

Therefore, if we pick k s.t. we keep 99% of the variance in our data, it means that the ratio of the projection errors to the squared distances of our data set is 1%.

Andrew Ng claims that there are many real-life examples where we can retain $\geq$ 95% of variance & considerably reduce the dimension of the data set.

Lecture 88: Advice for Applying PCA

PCA can be used to speed up learning algorithms. Consider a supervised learning algorithm, where our training set is $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$ & $x^{(i)} \in \mathbb{R}^{10,000}$. An example of such case could be an image recognition

problem, where the input data consists of 100 px × 100 px images.
In such a case, the learning algorithm may be super slow, just because there are so many features. We can use PCA:

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

$$\downarrow \text{PCA}$$

$$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$$

and then train the following data set: $(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})$.
How about predicting an unseen example? $x \xrightarrow[\text{PCA}]{} z \xrightarrow{h_\theta(z)}$ prediction.
Note that we should run PCA on the training set, and use the results to map examples in cross-validation & test sets. In other words, we should diagonalize $\hat{X}_{\text{training}}^T \hat{X}_{\text{training}}$ to get U_{training} & use this reduce the dimension of cross-validation & test-set examples:

$$x_{cv}^{(i)} \xrightarrow{U_{\text{training}}} z_{cv}^{(i)}$$

$$x_{\text{test}}^{(i)} \xrightarrow{U_{\text{training}}} z_{\text{test}}^{(i)}$$

Another note of caution: PCA shouldn't be used to prevent overfitting. When there are a lot of features, overfitting may happen, so it's tempting to think that reducing the # of features is a good way of preventing overfitting. Although this might work in some cases, regularization is a

better way of addressing overfitting. One reason is that PCA knows nothing about the targets $y^{(i)}$, while regularization does.

Yet another note of caution: always try to run your algorithm with the original data set first. Only if that doesn't do what you want, for instance if it's too slow, try reducing the dimension using PCA.