March 26, 2018

CS 361: Probability & Statistics

Multidimensional data

Example problem

Suppose I had a large dataset of 64x64 grayscale pictures of faces

Thus, each point in the dataset is a 4096 dimensional vector.

We are going to figure out a way to store the same images with high accuracy but using only 16 dimensions per image

Dealing with high dimensional data

High dimensional data

We will be supposing that we have a dataset $\{\mathbf{x}\}$ of N data items, with each point being d numerical dimensions. We will thus think of our data items as vectors, which can be added, subtracted, and multiplied by constants

When we want to indicate the vector corresponding to the i-th data item, we will write $\mathbf{x}_i$

If we want to refer to the j-th component of the i-th data item, we will write $\mathbf{x}_i^{(j)}$

The mean

Just as in one dimension, we can have a mean for a dataset. In this context it makes sense to talk about the **mean vector** and calculate it as

$$\text{mean}\left(\{\mathbf{x}\}\right) = \frac{\sum_i \mathbf{x}_i}{N}$$

Where the j-th component of the mean vector is given by

$$\text{mean}(\{x\})^{(j)} = \frac{\sum_i x_i^{(j)}}{N}$$

Just as with the 1 dimensional mean we have $\text{mean}\left(\{\mathbf{x} - \text{mean}\left(\{\mathbf{x}\}\right)\}\right) = 0$ which is to say, we can produce a 0 mean dataset by subtracting the mean from each data item

Blobs

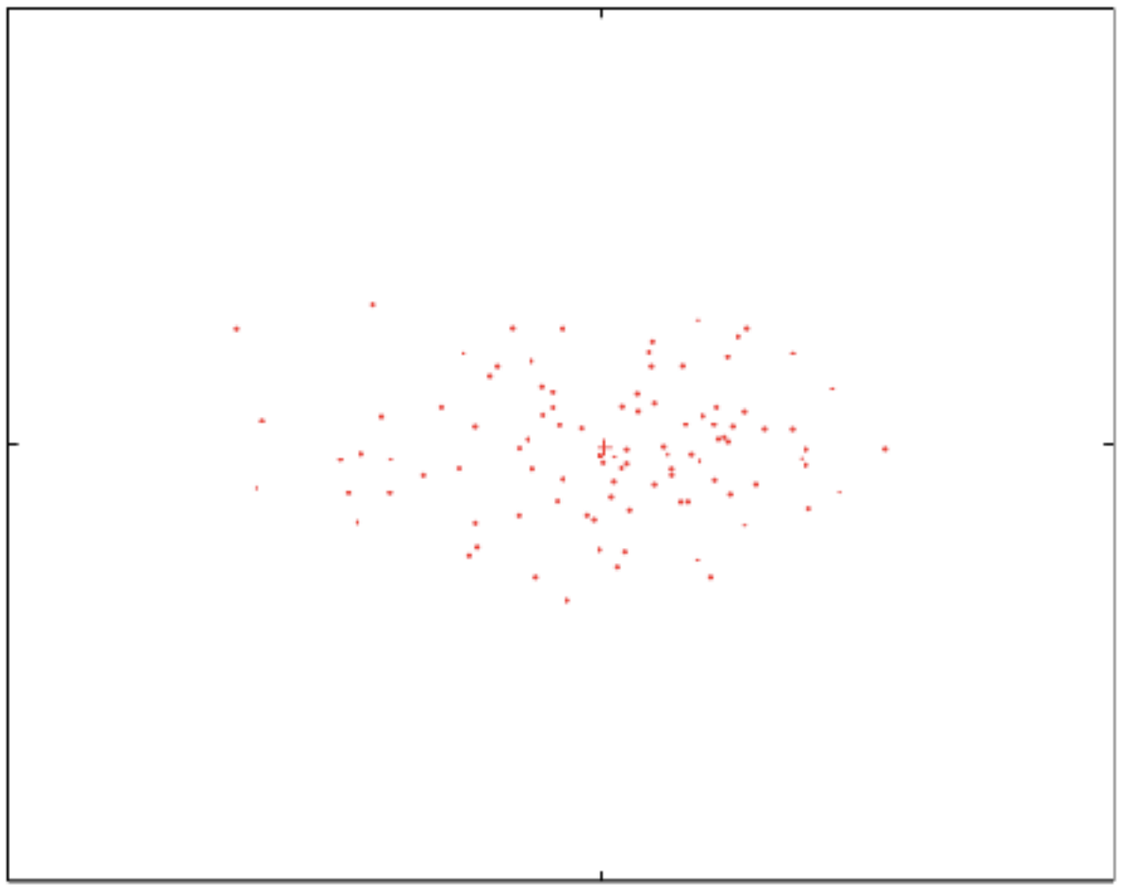
When we had 1d data and plotted histograms, when the data had one mode, the mean and variance were quite useful to us as summaries of the data

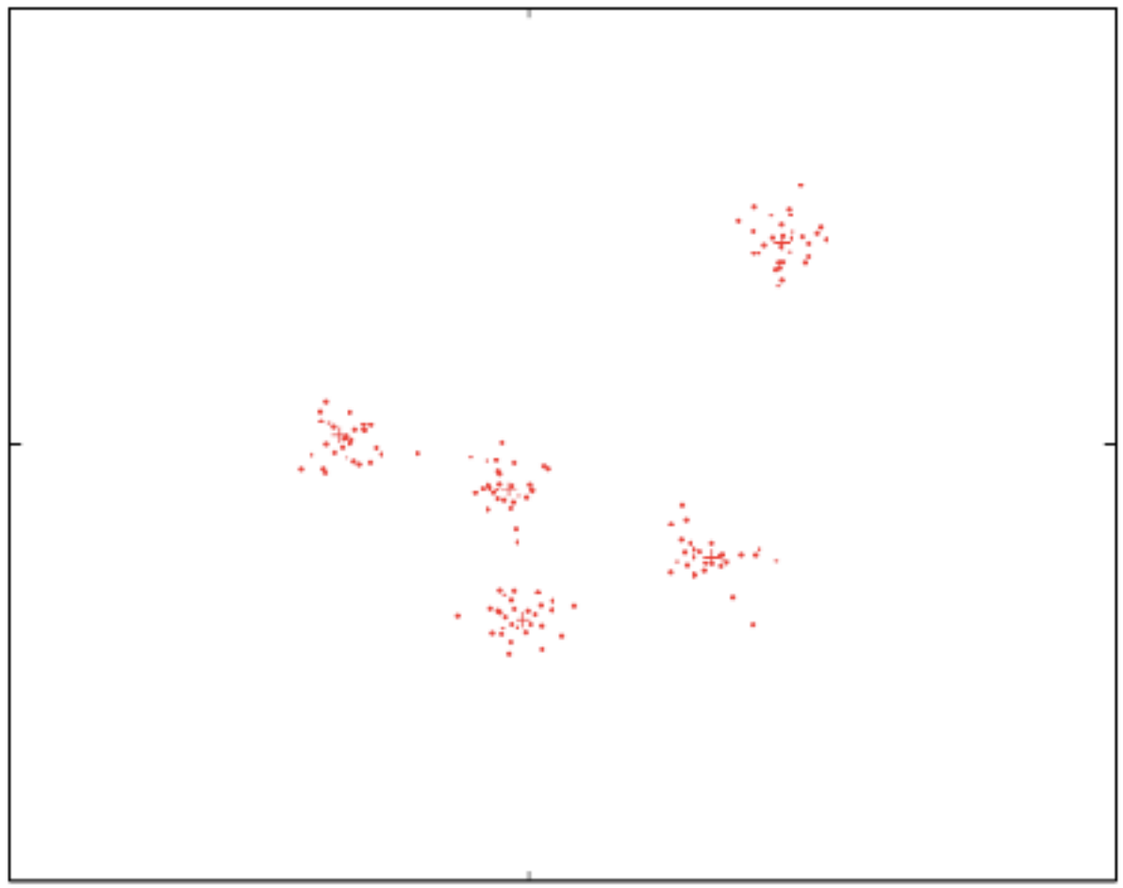
A high dimensional analogue of a unimodal histogram is a blob—a group of data points that clusters nicely together

Sometimes data forms multiple clumps or blobs and we will learn how to think about those in a few chapters

When we have a single blob, the mean and covariance—which will explore in more depth later— are useful for summarizing data

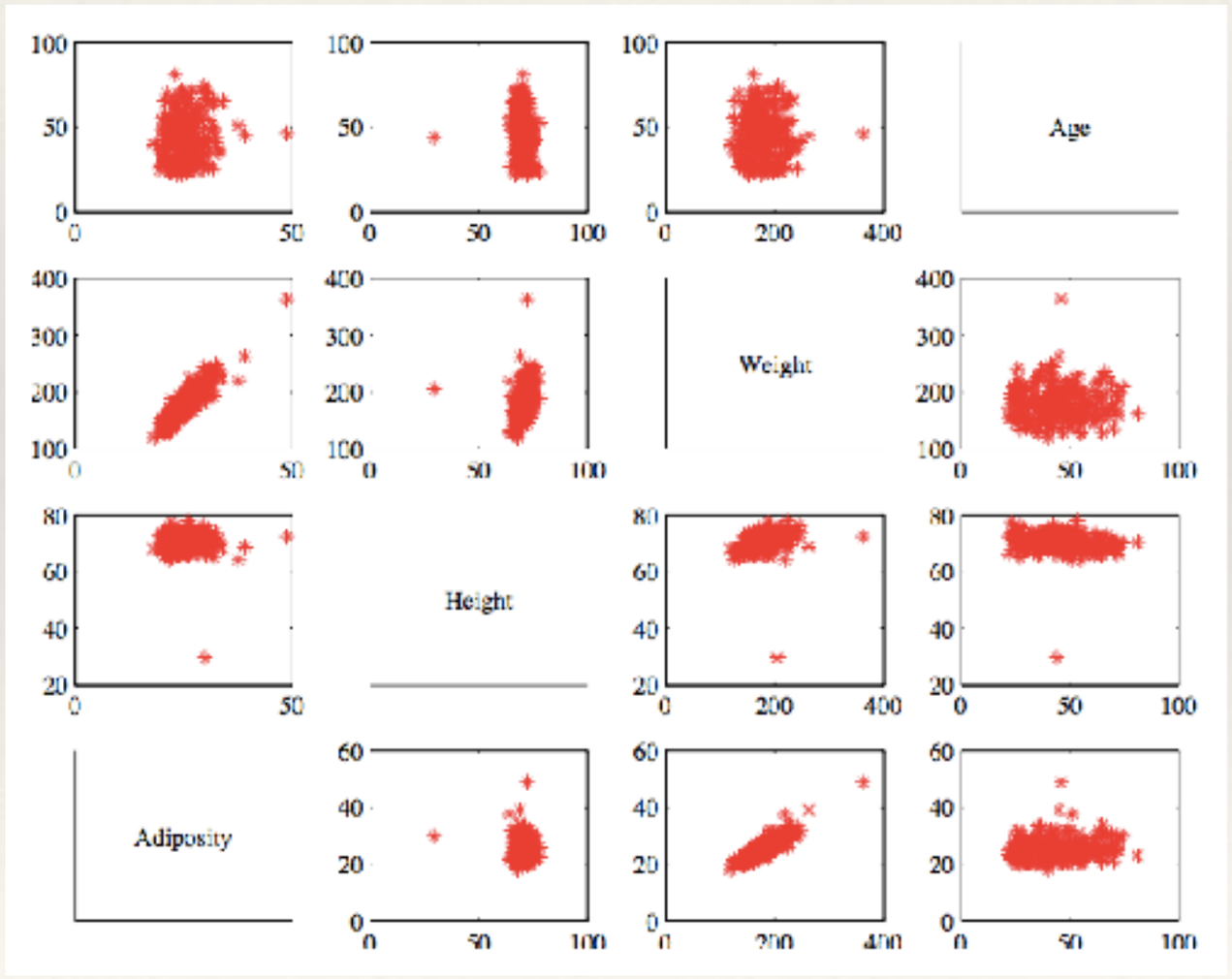
Blobs





Visualizing high dimensional data

Plotting high dimensional data is tricky. If there are greater than 3 dimensions we clearly can't just plot the data. If there aren't too many dimensions, we might get some insight by plotting a so-called scatterplot matrix. Here is a 4 dimensional dataset



Covariance

If we have two datasets {x} and {y} of the same number of items, the dataset covariance between x and y is given by the following

Definition: 10.1 *Covariance*

Assume we have two sets of N data items, $\{x\}$ and $\{y\}$. We compute the covariance by

$$\text{cov}(\{x\},\{y\}) = \frac{\sum_i (x_i - \text{mean}(\{x\}))(y_i - \text{mean}(\{y\}))}{N}$$

Example

A dataset of heights and weights is as follows where height is the first coordinate and weight the second: {(67.5, 154.25), (72.25, 173.25), (66.25, 154.00), (64.75, 133.25)}. What is the covariance of the height and weight?

First we get the mean of both heigh and weight: 67.6875 and 153.6875 respectively

Then we compute the covariance as:
(67.5-67.6875)(154-153.6875)/4 + (72.25-67.6875)(173.25-153.6875)/4 +
(66.25-67.6875)(154.00-153.6875)/4 + (64.75-67.6875)(133.25-153.6875)/4

And get a result of 37.195

Covariance

The “datasets” we are using with covariance are often two of the dimensions of a single multi-dimensional dataset

Notice that in the definition of covariance the order of the data matters.

A high covariance happens when item *i* tends to be larger than its mean in both datasets {x} and {y} at the same time and a large negative covariance occurs when {x} and {y} are far from the mean in opposite directions for datapoint *i*—i.e. one is larger than its mean when the other is smaller and vice versa

Covariance

Notice that

$$\text{std}\left(x\right)^2 = \text{var}\left(\{x\}\right) = \text{cov}\left(\{x\}, \{x\}\right)$$

And, a concept from Chapter 2

$$\text{corr}\left(\{(x,y)\}\right) = \frac{\text{cov}\left(\{x\}, \{y\}\right)}{\sqrt{\text{cov}\left(\{x\}, \{x\}\right)}\sqrt{\text{cov}\left(\{y\}, \{y\}\right)}}.$$

which gives a useful way to think about correlation. It measures the tendency of {x} and {y} to be large (or small) together relative to how much x and y themselves vary

Covariance matrix

We can form a matrix to represent the covariance between all of the d dimensions of a dataset

Definition: 10.2 *Covariance Matrix*

The covariance matrix is:

$$\text{Covmat}(\{\mathbf{x}\}) = \frac{\sum_i (\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))^T}{N}$$

Notice that it is quite usual to write a covariance matrix as Σ, and we will follow this convention.

We may refer to this covariance matrix as $\boldsymbol{\Sigma}$

Covariance matrix

The j, k-th entry of the covariance matrix is the covariance of the j-th and k-th dimensions of the data

The j, j-th entry of the covariance matrix is the variance of the j-th dimension of the dataset

The covariance matrix is symmetric

Proofs

Proposition: The j, k-th entry of the covariance matrix is the covariance of the j-th and k-th dimensions of the data

Proof: from the definition

$$\text{Covmat}\left(\{\mathbf{x}\}\right) = \frac{\sum_i (\mathbf{x}_i - \text{mean}\left(\{\mathbf{x}\}\right))(\mathbf{x}_i - \text{mean}\left(\{\mathbf{x}\}\right))^T}{N}$$

The j, k-th entry of this matrix is

$$\frac{\sum_i (x_i^{(j)} - \text{mean}\left(\{x^{(j)}\}\right))(x_i^{(k)} - \text{mean}\left(\{x^{(k)}\}\right))^T}{N}$$

Which is the covariance of dimension j and k of the data

Proofs

Proposition: The j, j-th entry of the covariance matrix is the variance of the j-th dimension of the dataset

Proof: from the last result, we have

$$\text{Covmat}\left(\{\mathbf{x}\}\right)_{jj} = \text{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(j)}\right\}\right)$$

And we observed earlier that this is the variance of component j

Proofs

Proposition: The covariance matrix is symmetric

Proof: Another way of stating this is to say that

$$\text{Covmat}\left(\{\mathbf{x}\}\right) = \text{Covmat}\left(\{\mathbf{x}\}\right)^T$$

But we have

$$\begin{aligned}
\text{Covmat}\left(\{\mathbf{x}\}\right)_{jk} &= \text{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(k)}\right\}\right) \\
&= \text{cov}\left(\left\{x^{(k)}\right\}, \left\{x^{(j)}\right\}\right) \\
&= \text{Covmat}\left(\{\mathbf{x}\}\right)_{kj}
\end{aligned}$$

Summary

With a 1 dimensional dataset, two scalar quantities—the mean and the variance—were useful for summarizing the data. One gave us a sense of the location of the data, the other a sense of its scale or spread

In the case of multi-dimensional data, we need a vector—called the mean—to specify a meaningful location, and a matrix—the covariance matrix—to characterize the spread of the data

Affine transformations

Transformations

Why would we transform data? We did this for single dimensional data when we converted a dataset to standard normal coordinates. We might like to transform a blob to make it easier to study.

It also turns out that our trick for representing 64x64 images with just a few dimensions will involve doing a specific transformation

What does transforming a blob consist of? Perhaps transforming each item with a matrix and then adding a constant vector to each item

Covariance and means are so useful to describing data that we want to keep track of what happens to them for an when we transform the data

Transformations

Suppose we have a dataset $\{\mathbf{x}\}$ and we transform it to a new dataset $\{\mathbf{u}\}$ according to the following with some matrix A and vector $\boldsymbol{b}$

$$\mathbf{u}_i = A\mathbf{x}_i + \mathbf{b}$$

(this is called an affine transformation—a linear transformation then adding a vector)

What will be the mean and covariance of the new dataset $\{\mathbf{u}\}$?

Transformations: mean

$$\text{mean}(\{\mathbf{u}\}) = \text{mean}(\{\mathcal{A}\mathbf{x} + \mathbf{b}\})$$

Using the definition of mean we get

$$\text{mean}(\{\mathbf{u}\}) = \frac{\sum_i \mathbf{u}_i}{N}$$

With the linearity of summation

$$\text{mean}(\{\mathbf{u}\}) = \mathcal{A}\sum_i \frac{\mathbf{x_i}}{N} + b$$

Using our transformation we get

$$\text{mean}(\{\mathbf{u}\}) = \frac{\sum_i (\mathcal{A}\mathbf{x_i} + \mathbf{b})}{N}$$

And using the definition of the mean

$$\text{mean}(\{\mathbf{u}\}) = \mathcal{A}\,\text{mean}(\{\mathbf{x}\}) + \mathbf{b}$$

Transformations: covariance

Let's calculate

$$\text{Covmat}(\mathbf{u}) = \text{Covmat}(\{\mathcal{A}\mathbf{x} + \mathbf{b}\})$$

By the definition of covariance matrix

$$\text{Covmat}(\mathbf{u}) = \frac{\Sigma_i(\mathbf{u}_i - \text{mean}(\{\mathbf{u}\}))(\mathbf{u}_i - \text{mean}(\{\mathbf{u}\}))^T}{N}$$

Substituting with our transformation

$$\text{Covmat}(\{\mathbf{u}\}) = \frac{\Sigma_i(\mathcal{A}\mathbf{x}_i + \cancel{\mathbf{b}} - \mathcal{A}\text{mean}(\{\mathbf{x}\}) - \cancel{\mathbf{b}})(\mathcal{A}\mathbf{x}_i + \cancel{\mathbf{b}} - \mathcal{A}\text{mean}(\{\mathbf{x}\}) - \cancel{\mathbf{b}})^T}{N}$$

Transformations: covariance

Using the fact that A is a linear transformation

$$\text{Covmat}(\{\mathbf{u}\}) = \frac{\mathcal{A}\Sigma_i(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))^T\mathcal{A}^T}{N}$$

Using the definition of the covariance matrix

$$\text{Covmat}(\{\mathbf{u}\}) = \mathcal{A}\text{Covmat}(\{\mathbf{x}\})\mathcal{A}^T$$

Summary

So now we know that if we transform our dataset by hitting each datapoint with a matrix A and adding to each a constant vector b

$$\mathbf{u}_i = \mathcal{A}\mathbf{x}_i + \mathbf{b}$$

We have the following two results:

$$\text{mean}(\{\mathbf{u}\}) = \mathcal{A}\text{mean}(\{\mathbf{x}\}) + \mathbf{b}$$

$$\text{Covmat}(\{\mathbf{u}\}) = \mathcal{A}\text{Covmat}(\{\mathbf{x}\})\mathcal{A}^T$$

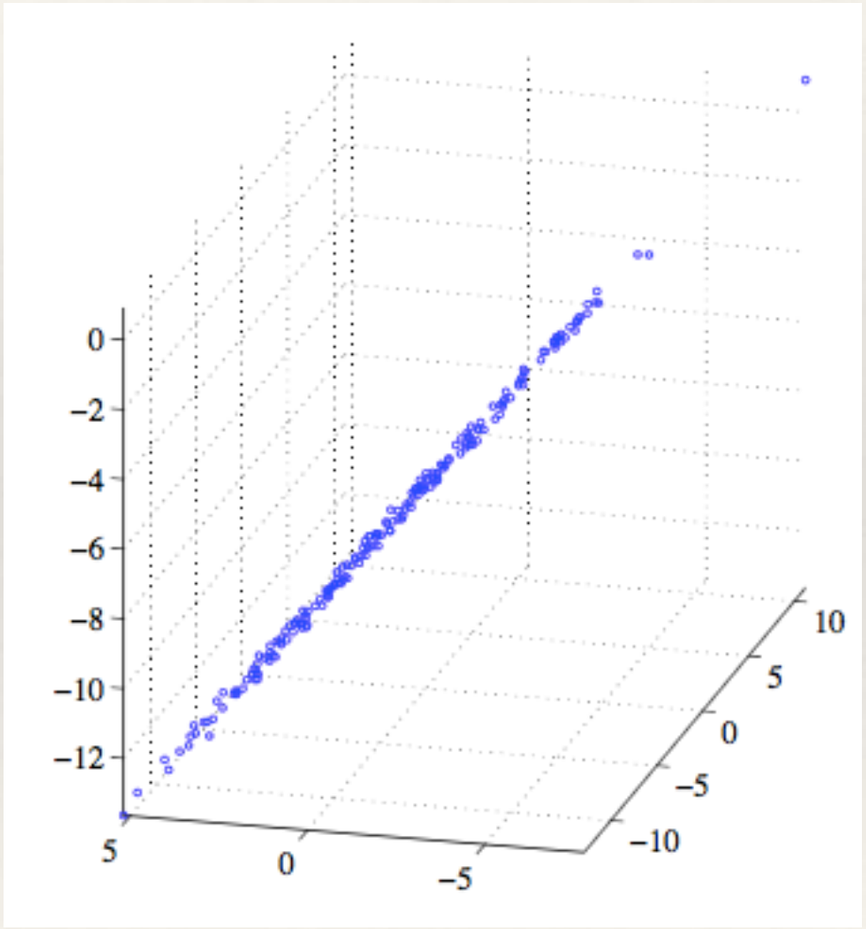
A special transformation

Transformations: subspaces

There are at least a few reasons why we might want to transform our dataset that we will look at here

Our d-dimensional data might in practice lie approximately on some linear surface (a hyperplane) of lower dimension

Though the data pictured is from a dataset with 3 dimensions, it might not be unwise at all to treat it as though it were 1 dimensional if we could get an axis to run parallel to the data



Correlated components

The motion of a spring could be measured with (x,y,z) coordinates, but from physics we know we actually only need one quantity to describe the motion of a spring over time. If we have (x,y,z) data, can we recover this 1 dimensional characterization of the dataset and express the data in terms of this one coordinate?

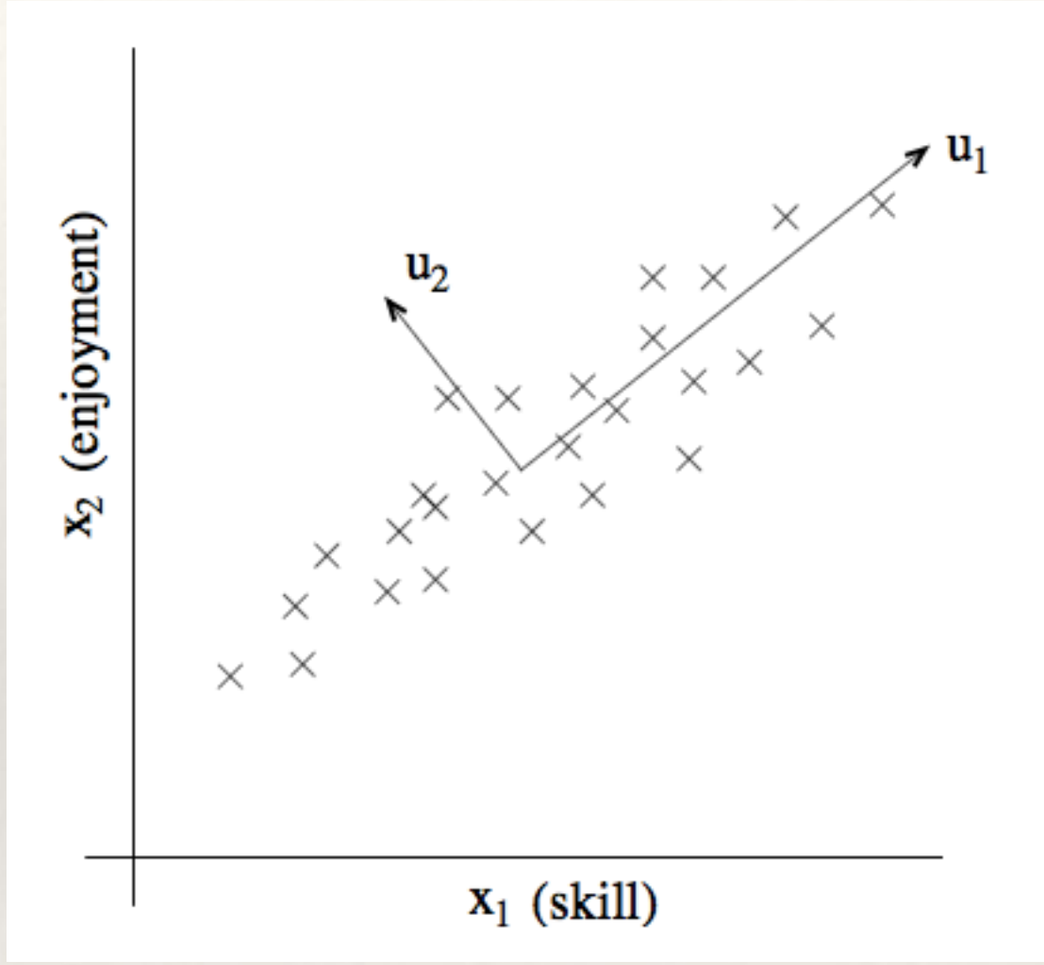
If we had a large dataset of people and the ratings they give the movies they have seen we expect to see some correlation in the ratings due to the fact that movie genres exist. There may be a correlation between liking Justice League and The Avengers, for example.

If a dataset generated from looking at server logs and the response time, request load, available memory, and CPU utilization were all recorded for each server request, we are probably going to see some correlation in these dimensions

Correlated components

If our data has correlated dimensions we should be able to find some coordinate system wherein the axes of the coordinates line up with the directions of maximum variance in the data

We might be able to discard the components that don't have much variation in their direction (maybe we would just store the u1 coordinates to the right and not the u2)



Source: Andrew Ng machine learning notes

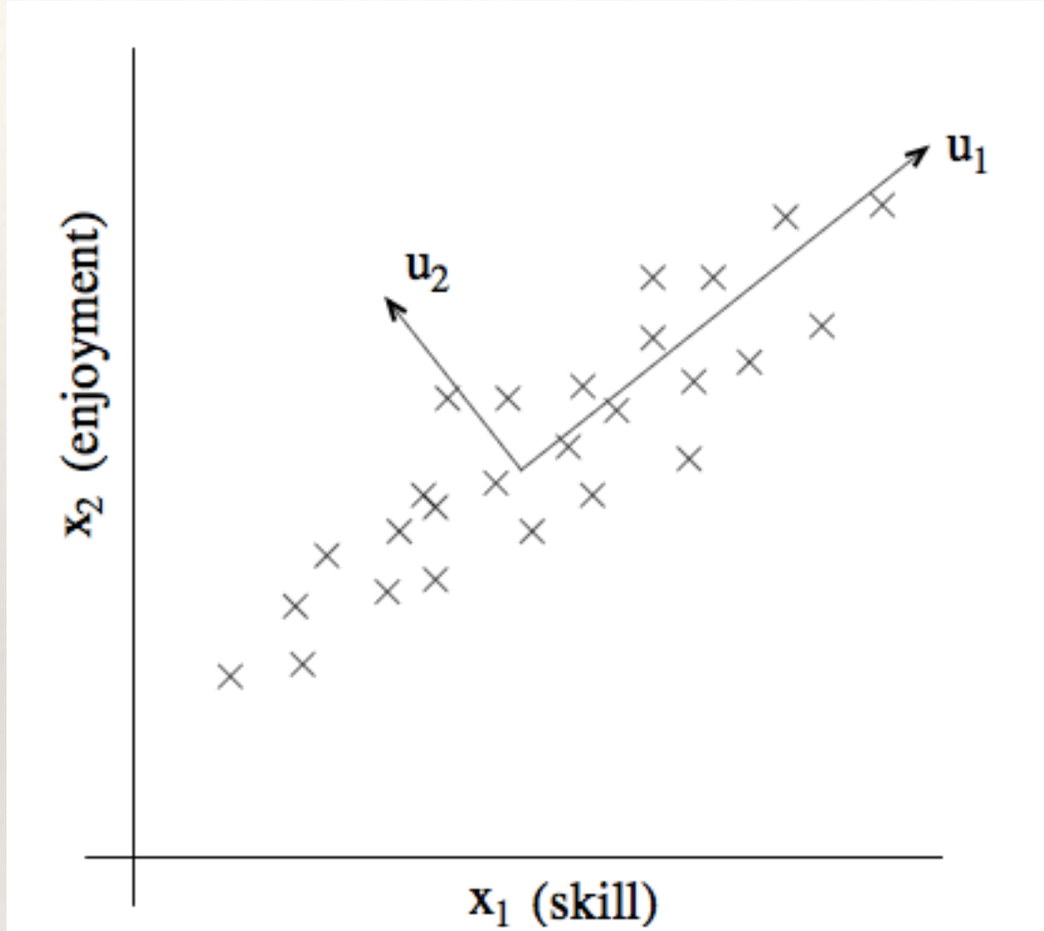
We would lose some interpretability if we can't describe exactly what u1 means but it might be worth it to cut down the amount of data we have to store

A useful transformation

We are going to see what kind of transformation we would have to do to our data to get coordinates like these

It will be a transformation which gives us data with 0 mean

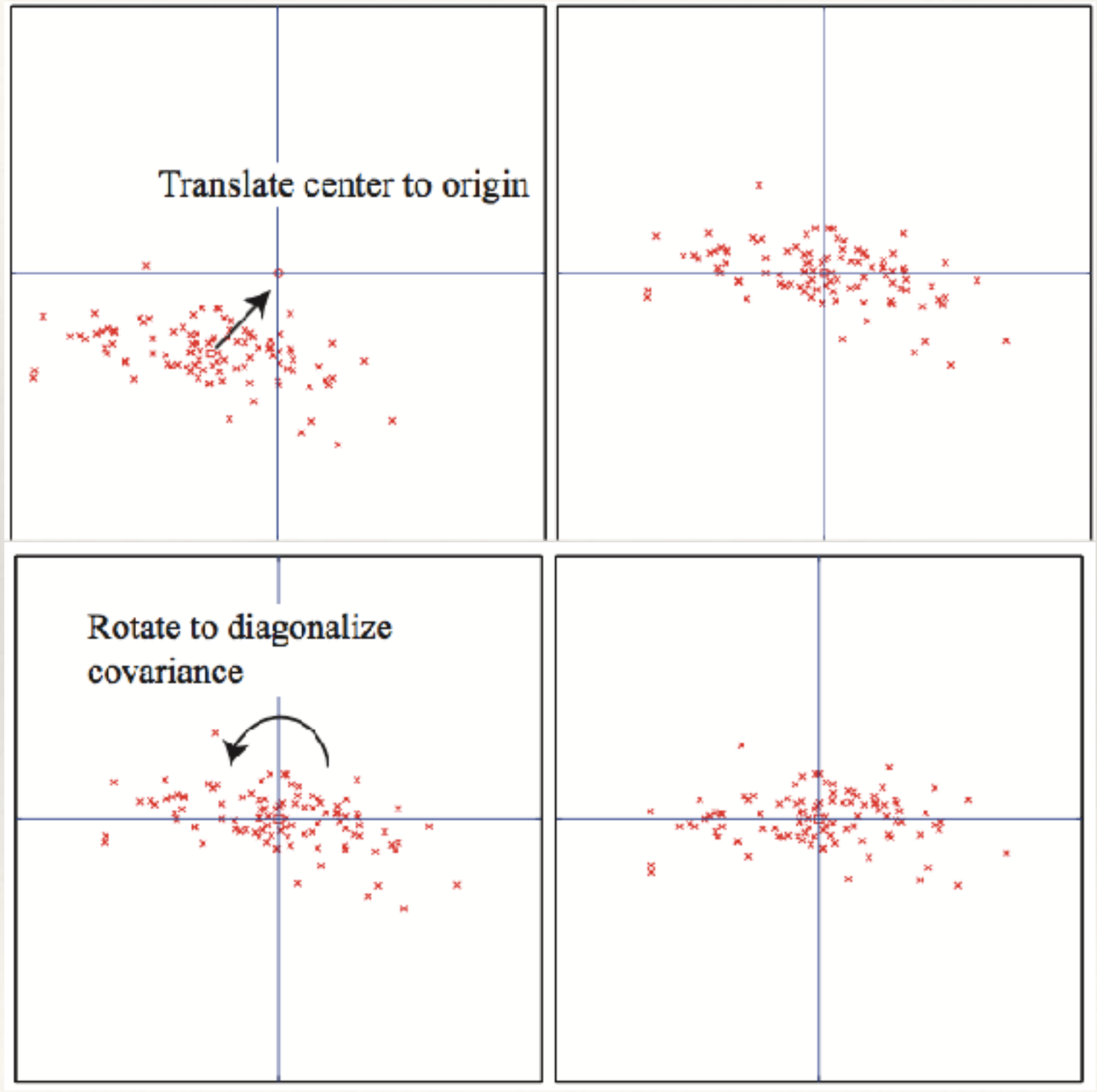
It will be a transformation which gives us data with 0 covariance between dimensions



Source: Andrew Ng machine learning notes

A translation and a rotation

Our transformation will consist of a translation followed by a rotation of the data



A reminder from linear algebra

Earlier we showed that the covariance matrix of the data is symmetric. A result you may or may not recall from linear algebra is that a symmetric matrix can be diagonalized. Which means we can write the covariance matrix as

$$\text{Covmat}(\{\mathbf{x}\}) = U \Lambda U^T$$

Where U is a matrix of orthonormal eigenvectors of the covariance matrix. And Λ is a diagonal matrix with entries corresponding to the eigenvalues of the covariance matrix

It's also the case that the transpose of an orthonormal matrix is its inverse, i.e.

$$UU^T = U^T U = I$$

The transformation

Now if we write $\mathbf{u}_i = \mathbf{x}_i - \text{mean}(\{\mathbf{x}\})$ we will have a dataset with a mean of 0. And if we then transform that dataset with the matrix of eigenvectors of $\{\mathbf{x}\}$

$$\mathbf{n}_i = \mathcal{U}^T \mathbf{u}_i$$

We will have a transformation worth looking at. Now, the mean of the dataset is 0. And our result from a few slides ago about covariances, tells us that the covariance of this dataset is

$$\text{Covmat}(\{\mathbf{n}\}) = \mathcal{U}^T \text{Covmat}(\{\mathbf{x}\}) \mathcal{U}$$

But we just said on the last slide that

$$\text{Covmat}(\{\mathbf{x}\}) = \mathcal{U} \Lambda \mathcal{U}^T$$

Or

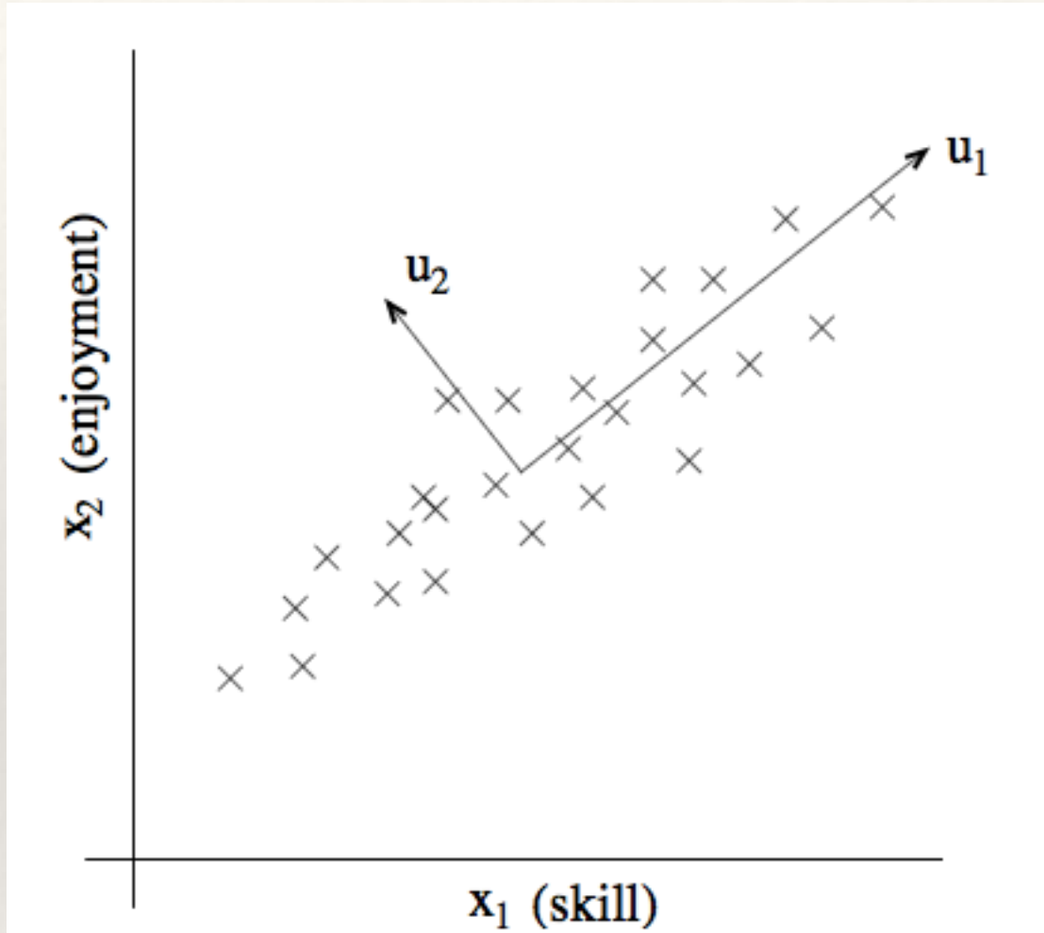
$$\text{Covmat}(\{\mathbf{n}\}) = \Lambda$$

How did we do?

We said we wanted a transformation.

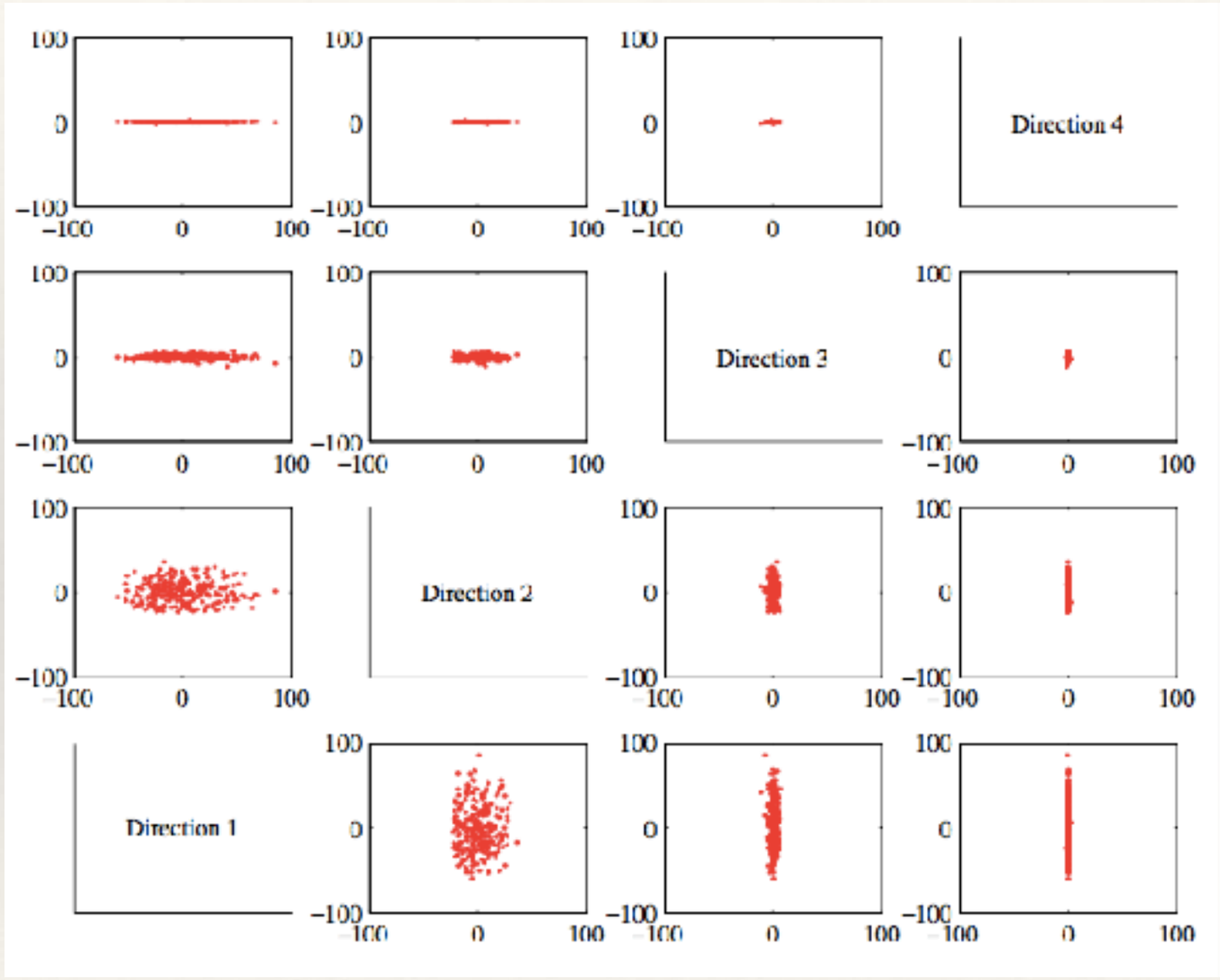
That will be a transformation which gives us data with 0 mean

That will be a transformation which gives us data with 0 covariance between dimensions



Source: Andrew Ng machine learning notes

Ordering the eigenvalues



Example

This process will give us the ability to construct a new dataset by giving us a matrix of eigenvectors U and a matrix of eigenvalues. How should we interpret these matrices?

$$\text{Covmat}(\{\mathbf{x}\}) = \mathcal{U}\Lambda\mathcal{U}^T$$

Suppose we looked at data on countries in the world and the data had dimensions for population, area, GDP, infant mortality, and life expectancy. Suppose that we found the eigenvalues and eigenvectors of the covariance matrix of this data and got for the first two components

$$u_1 = \begin{bmatrix} 0.931 \\ -0.351 \\ -0.096 \\ -0.004 \\ -0.008 \end{bmatrix} \quad \Lambda_1 = 23973.32 \qquad u_2 = \begin{bmatrix} -0.338 \\ -0.931 \\ 0.125 \\ 0.030 \\ 0.041 \end{bmatrix} \quad \Lambda_2 = 16949.71$$

PCA dimensionality reduction

So we have created a new dataset that is still d dimensional and has mean 0, uncorrelated dimensions, and dimensions that have decreasing variance

$$\mathbf{u}_i = \mathbf{x}_i - \text{mean}(\{\mathbf{x}\})$$

$$\mathbf{n}_i = \mathcal{U}^T \mathbf{u}_i$$

The variance in the first component is the largest eigenvalue of Covmat({x}), the variance in the second component is the second largest eigenvalue of Covmat({x}) and so-on

Which means we might get a good approximation of the dataset in only *r* dimensions if we only keep around the first *r* components of **n**. These are the components with the highest variance in our transformation, after all