**Principal Component Analysis(PCA) problem formulation**

e.g.: Reduce from 2-dimension to 1-dimension: Find a direction(a vector u\_1 є Rn) onto which to project the data, so as to minimize the projection error.

Generally, Reduce from n-dimension to k-dimension: Find k vectors u\_1, u\_2, u\_3, …, u\_k onto which to project the data, so as to minimize the projection error.

Before applying PCA, there is a data preprocessing step.

Training set: x\_1, x\_2, x\_3, …, x\_m.

Preprocessing(feature scaling/mean normalization):

μj = 1/m \* sum(xij)(i from 1 up to m)

replace each xij with xj – uj

If different features on different scale(e.g., x1 = size of house, x2 = number of bedrooms), scale features to have a comparable range of values.

**PCA Algorithm:**

Reduce data from n-dimensions to k-dimensions

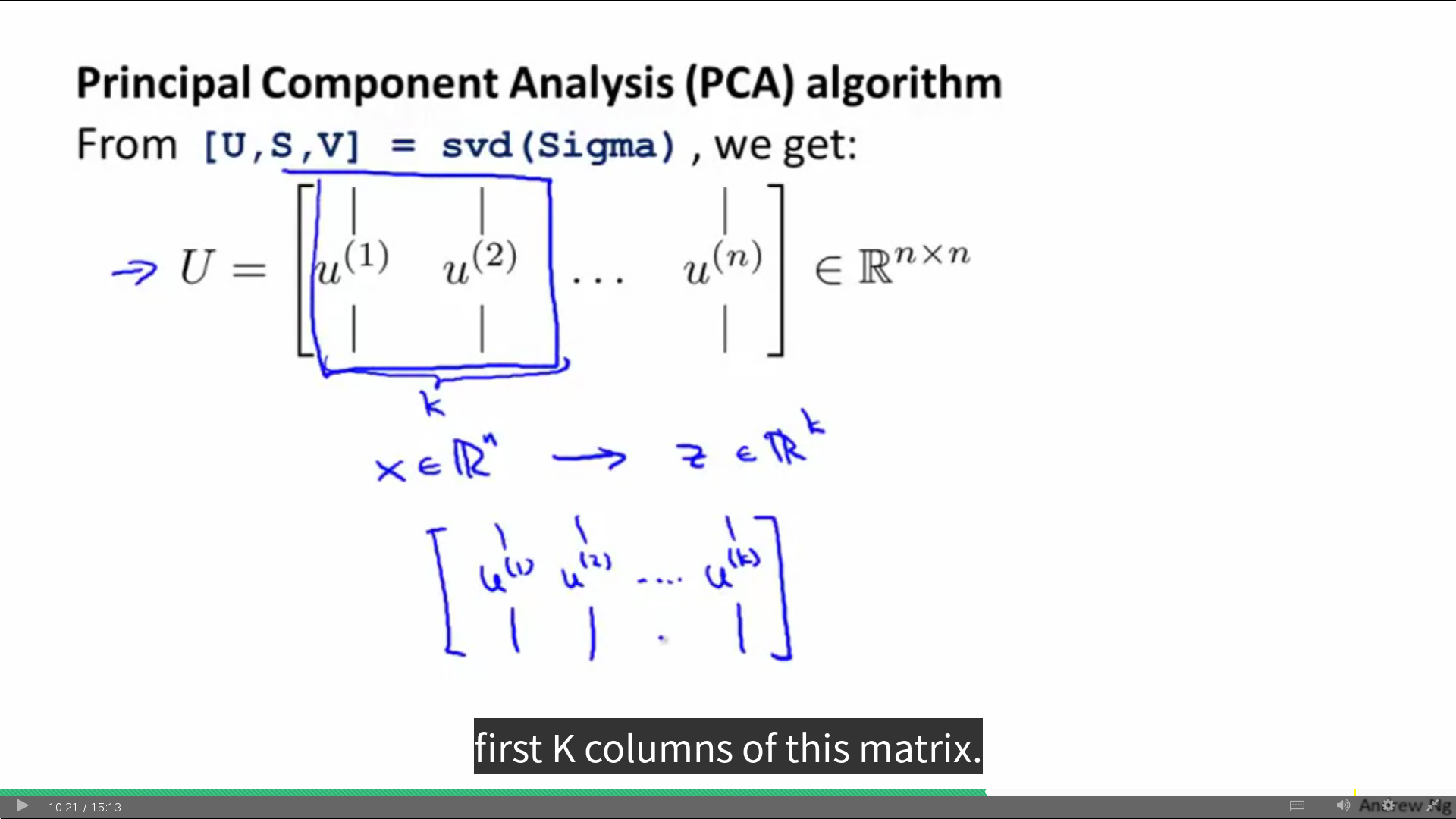
Compute “covariance matrix”:

∑ = 1/m \* sum((xi) \* (xi)T)(i = 1 through m)(∑ is a n \* n matrix)

Compute “eigenvectors” of matrix ∑:

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

After that, What we really need from svd function is the U matrix(it also be n \* n matrix). If we want to reduce n-dimensions to k-dimensions, we just need the first k column vectors of U.



After picking first k column vectors and stacking it into a new matrix called Ureduce., we will perform the follow:

z = (Ureduce)T \* x

x is n by 1 vector and (Ureduce)T is k by n matrix. So the z is k by 1 vector and it also exactly what we want.

Note: x here might be a training set/test set/cross validation set.

**Summary PCA Algorithm:**

After mean normalization(ensure every feature has zero mean) and optionally feature scaling:

Sigma = 1/m \* sum((xi) \* (xi)T)(i = 1 through m)(∑ is a n \* n matrix)

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

Ureduce = U(:, 1:k);

z = Ureduce' \* x;

**Reconstruction from Compressed Representation**

Given zi є Rk, to map zi to xi where xi є Rn.

xiapprox = Ureduce \* zi;(Ureduce is n by k matrix and zi is k by 1 vector), the xiapprox is n by 1 vector.

Note: For now, I think the PCA much more used in unlabeled data set.

How to choose k(number of principal components) parameter in PCA

Average squared projection error: 1/m \* sum(||xi – xiapprox||2)(i = 1 up to m)

Total variation in the data: 1/m \* sum(||xi||2)(i = 1 up to m)

Note: The goal of PCA Algorithm is to minimize the Average squared projection error.

Typically, choose k to be smallest value so that

Average squared projection error/Total variation in the data <= 0.01/0.05(1%/5%). (1)

“99%/95% of variance is retained”

**Choosing k(number of principal components) Algorithm:**

**Algorithm1:**

Try PCA with k = 1

Compute Ureduce, z1, z2, …, zm, x1approx, x2approx, …, xmapprox

Check if (1) <= 0.01?

update k until the (1) <= 0.01;

**Algorimth2:**

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

S is a diagonal matrix

the equation (1) can be rewirte by follow:

1 - sum(Sii)(i=1 up to k)/sum(Sii)(i = 1 up to n) <= 0.01

in other words:

sum(Sii)(i=1 up to k)/sum(Sii)(i = 1 up to n) >= 0.99

if you want to be sure that 99% of the variance is retained.

So what you can do is just slowly increase k(k=1,2,3...), to test the new expression to check whether it less than 0.01

Summary Choosing k

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

Pick smallest value of k for which

sum(Sii)(i=1 up to k)/sum(Sii)(i = 1 up to n) >= 0.99

(99%/95% of variance is retained)

Advice for Applying PCA

Supervised learning speedup

(x1,y1), (x2,y2), (x3,y3), …, (xm,ym)

Extract input:

Unlabeled data set: x1, x2, x3, …, xm є R10000

Applying PCA to produce z1, z2, z3, …, zm є R1000

New training set:

(z1,y1), (z2,y2), (z3,y3), …, (zm,ym)

Note: Mapping xi → zi should be defined by running PCA only on the training set(e.g.: Ureduce is learned by training set). After that, this mapping can be applied as well to the examples xicv and xitest in the cross validation and test sets.

**Bad use of PCA: To prevent overfitting**

**Use zi instead of xi to reduce the number of features to k < n.**

**Thus, fewer features, less likely to overfit.**

**This might work OK, but isn't a good way to address overfitting. Use regularization instead.**

**PCA is sometimes used where it shouldn't be**

**e.g.:**

**Design ML System:**

**Get training set**

**Run PCA to reduce xi in dimension to get zi.**

**Train logistic regression on new train set**

**Test on test set: Map xitest to zitest. Run h(z) on new test set.**

**Ask: How about doing the whole thing without using PCA.**

**Important:**

**Before implementing PCA, first try to running whatever you want to do with the original/raw data xi. Only if that doesn't do what you want, then implementing PCA and consider using zi.**