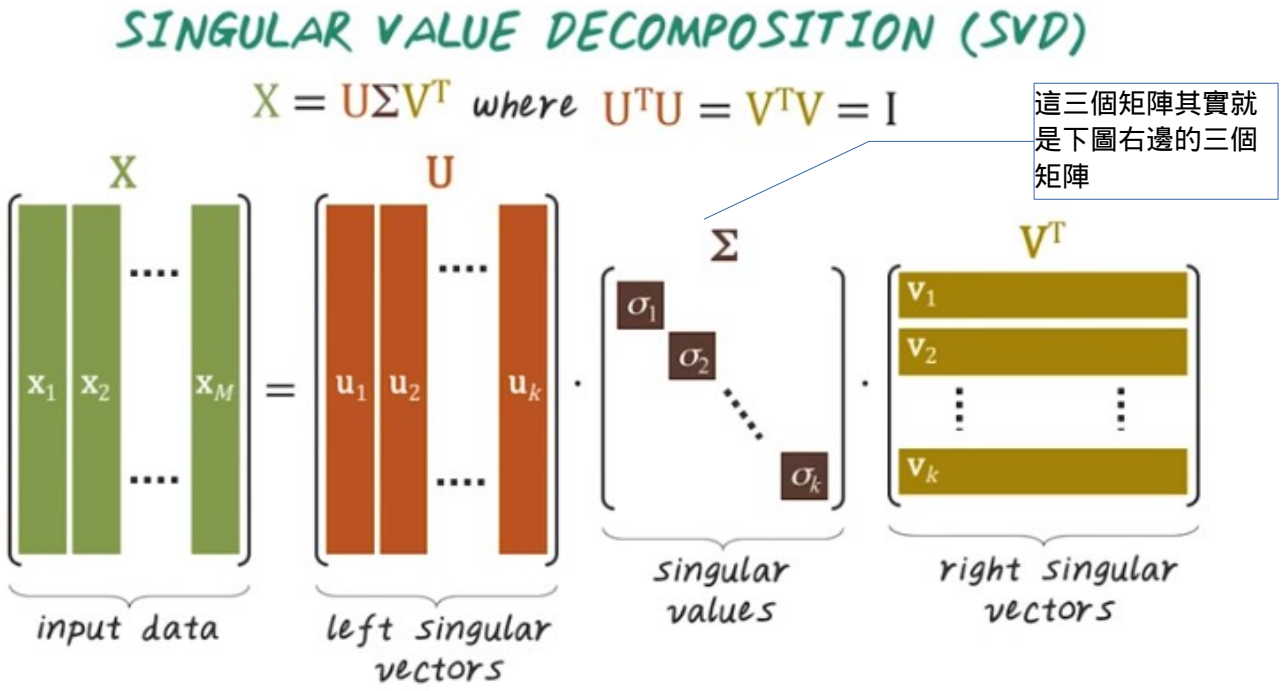
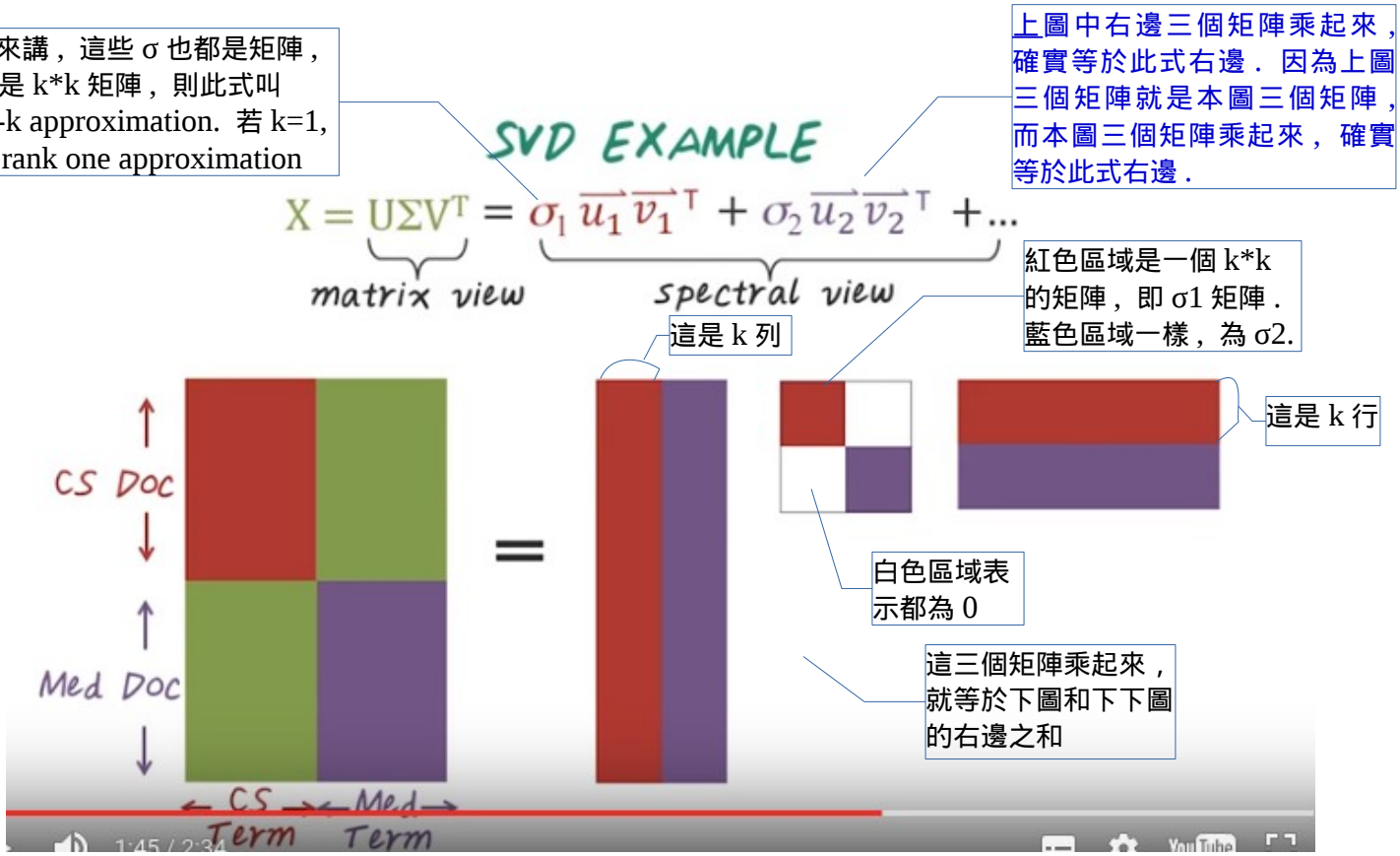
1. So now we're going to discuss dimensionality reduction. This is our method for distilling very complex and noisy raw data into robust core components. We'll start by talking about dimensionality reduction. Including singular value decomposition, principal component analysis, and CUR decomposition. Then, we'll discuss a more advanced method called Tensor Factorization. That handles higher order interaction among data. Finally, we'll see how this method applies to predictive modeling and phenotyping in healthcare.

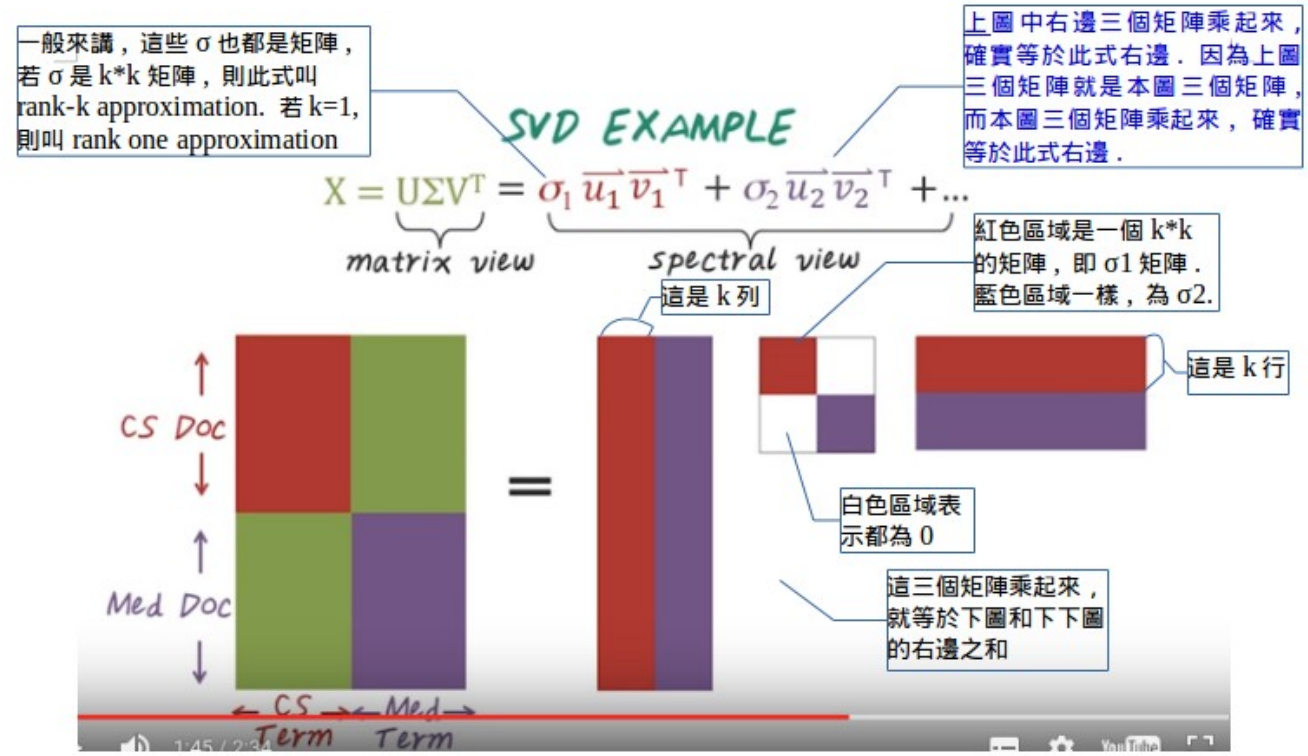
2. First, a recap of clustering algorithms. In previous lectures, we talked about hard clusterings, such as k-means, hierarchical clustering. We talked about soft clustering algorithms such as Gaussian mixture model. We also talked about scalable clustering algorithms. Such as, mini batch k-means and DBScan. In this lecture, we'll talk about dimensionality reduction. The reason for dimensionality reduction is because, it's often more robust and efficient to work with low dimensional data. Such as, a data set with 10 to 100 dimensions. But the original data, or the raw data often contains of much higher dimensional data set. Say, in order of tens of thousands to even a million dimensions. So, the dimensionality reduction is a set of method to reduce dimensionality of original data. While still maintaining the underlying data characteristics. So, we talk about Singular Value Decomposition, SVD, and Principal Component Analysis, PCA. Those are two classical method that summarize original set of features as linear combinations. Then we also talk about CUR decomposition. Instead of using linear combination of original features. CUR samples actual columns and rows from the original dataset. So, it's more intuitive and often leads to sparse result. Then we'll talk about tensor factorization, as another set of method to reduce dimensionality. When we're dealing with higher order tensor instead of matrixes.



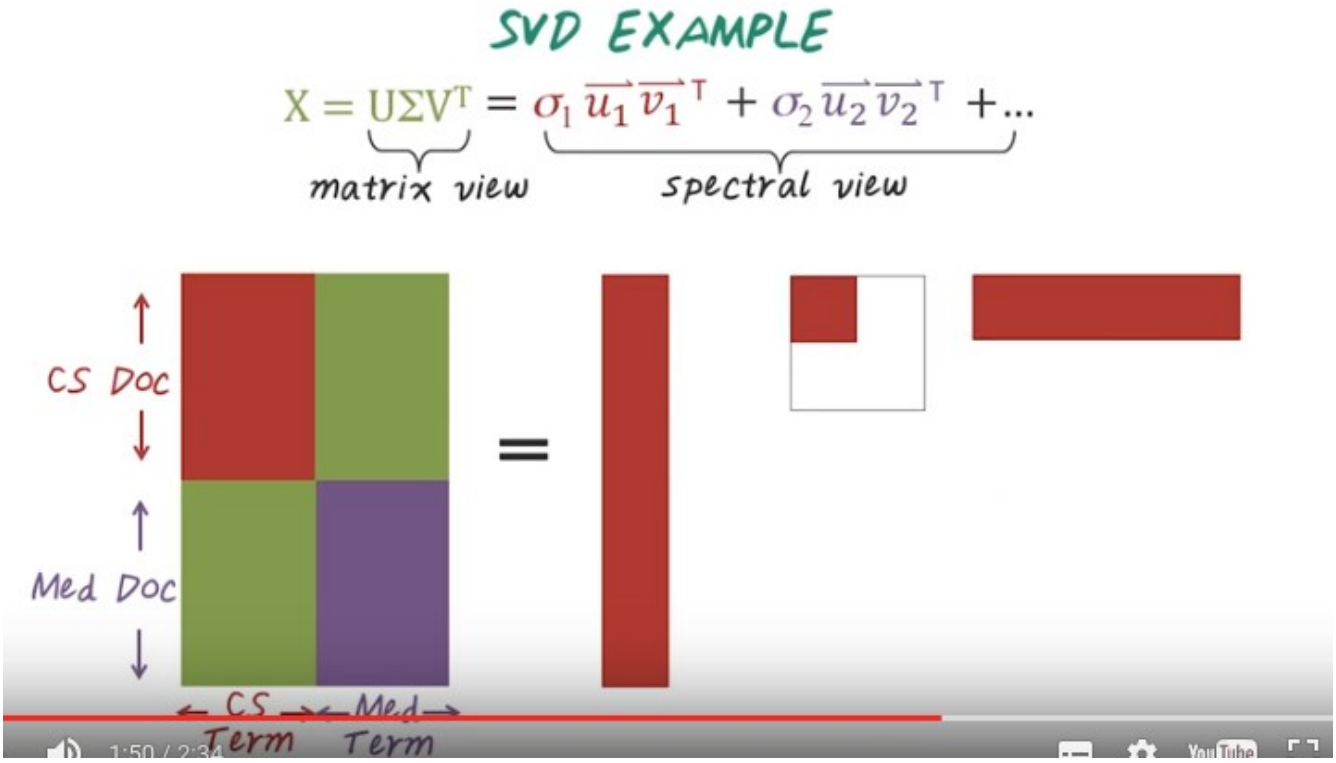
3. First, let's talk about singular value decomposition, or SVD. SVD factorizes the input matrix X as product of three matrices. U, Σ, V^T. Where the U and V matrices are which means U transpose times U equals to V transposed times V equal to identity matrix. Visually, given a large matrix X. And here, every columns represent a disease. Every row represent a patient. For example, X1 corresponding to the disease indicator of all the patient for the first disease and X2 represent the disease indicator for all the patient for the second disease, and so on. Then we can factorize X as matrix U times a diagonal matrix sigma times V transpose. Here X consists of all the input data, the patient by disease matrix, and the U matrix consists of left singular vectors. So every column here in U represent left singular vector. And sigma is a diagonal matrix, whereas diagonal elements are non-zeros, and off-diagonal elements are all zeros. And all the σs are the singular values. They're assorted in descending order. So sigma one is greater than or equal to sigma two, which is greater than or equal to sigma three, and so on. Then the columns in V corresponding to the right singular vectors. And this is the mathematical definition of singular vector decomposition.



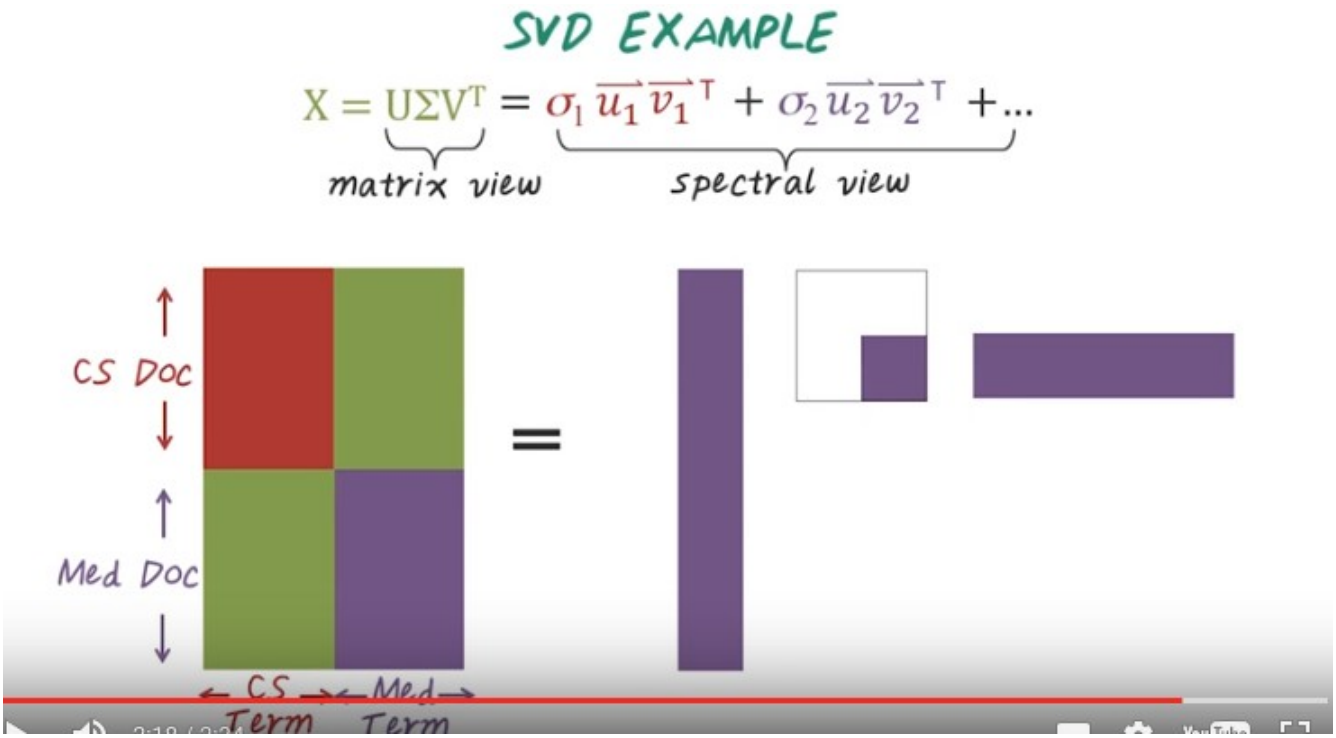
為了防止格式變亂, 又將上圖截圖成了下圖:



4. Now let's see an example of SVD. Given SVD over X, if we represent SVD as the matrix factorization, then we have this U times sigma times V transpose. However, we can also separate all those columns of U and V separately to obtain a different view we call spectral view. Here, sigma 1 times U1 times V1 transpose is a rank one approximation of the original matrix X, and sigma 2 U2 V2 transpose is another rank one approximation of the original matrix. If you sum up all those rank one approximations, you get original matrix X. So visually we have matrix X represent a set of documents and a set of terms used in those documents (例如: 縱坐標為{doc1, doc2, …}, 橫坐標為{term1, term2, …}, 一個元素表示例如 doc5 中九多少個 term3). For example here we have two set of documents. The red ones are computer science document and the purple ones are medical documents. And you can imagine the vocabulary used in these two sets of documents are very different. So the terms used in CS documents are very different from the terms used in the medical documents. If we want to summarize this big matrix using SVD, then we can summarize this input matrix as product of three matrices, U sigma V transpose. And if we look at the spectral view, what it means is, it's a set of rank one approximation.



For example, this red part corresponding to this first rank one approximation, so this is sigma 1 U1 V1 traspose. Putting them together gives us the CS documents and the corresponding terms.



Then we have another rank one approximation. This purple one's sigma 2, U2, V2 transpose. Multiply them together. Gives us the medical documents and terms. So by looking at the spectral view you can see that we reduce the [INAUDIBLE] of this huge matrix Into this two dimensional space. One for CS document, one for medical document.

SVD PROPERTIES

$$X = U\Sigma V^T$$

V are the eigenvectors of the covariance matrix X^TX

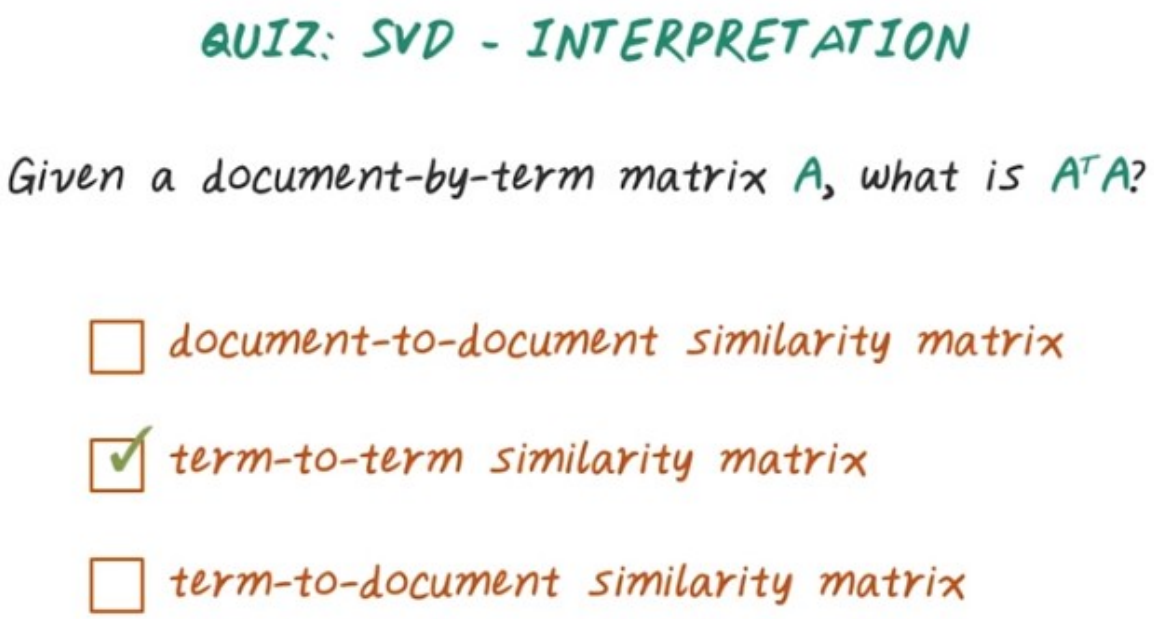
$$X^TX = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^2 V^T$$

U are the eigenvectors of the Gram (inner-product) matrix XX^T

$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma^2 U^T$$

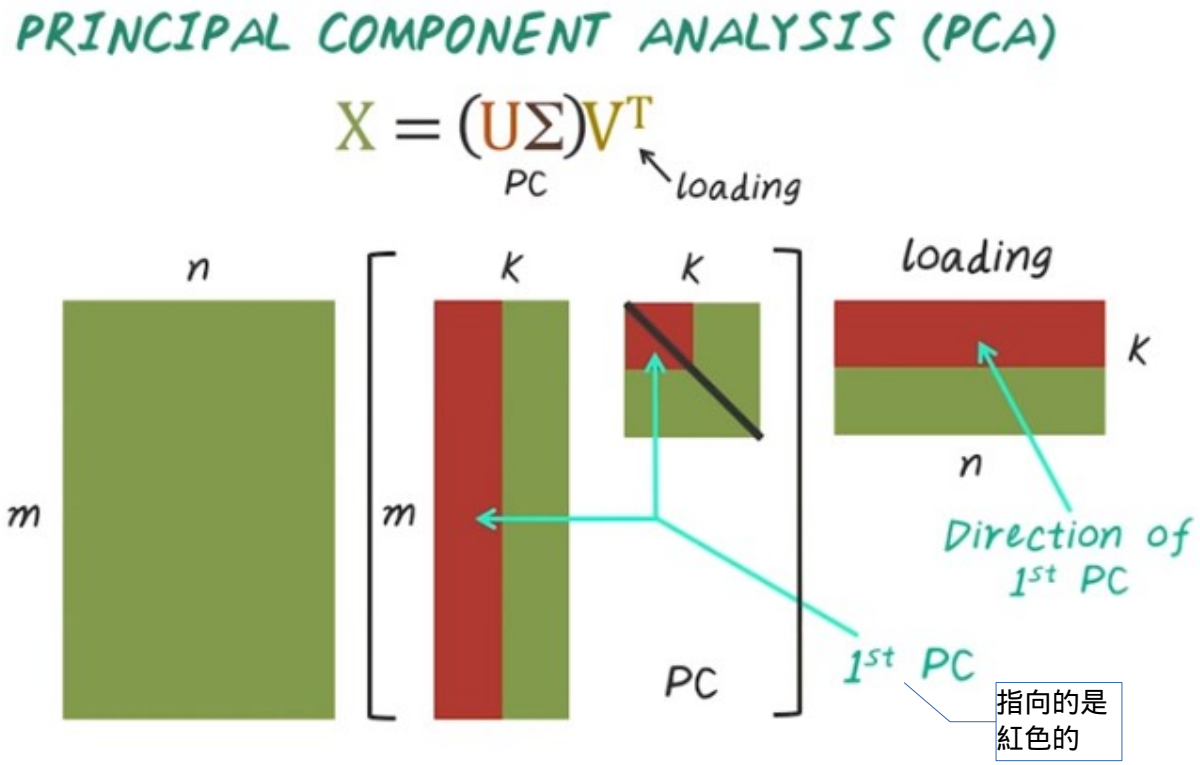
5. Next, let's talk about the property of SVDs. SVD is the matrix factorization, factorize input X as the product of three smaller matrices. U sigma, V transpose. And V are the eigenvectors of the covariance matrix X transpose X. So if we multiply X transpose X together and you carry out this calculation, you'll find out V times sigma square times V transpose equals X transpose X, which means V is the eigenvector of X transpose X. With eigenvalues Sigma square. And similarly, U are the eigenvectors of the Gram matrix, or inner-product matrix X X transpose. So here, given X X transpose, if you carry out this calculation, you find it equals U times Sigma square times U transpose. Which means U is the eigenvectors of X, X transpose with the eigenvalues sigma squared.

6. Let's do a quiz to understand SVD better. Given a document-by-term matrix like the one we have shown you in the previous slide, what is A transpose A? Is it a document-to-document similarity matrix? Or is it term-to-term similarity matrix? Or is term-to-document similarity matrix?

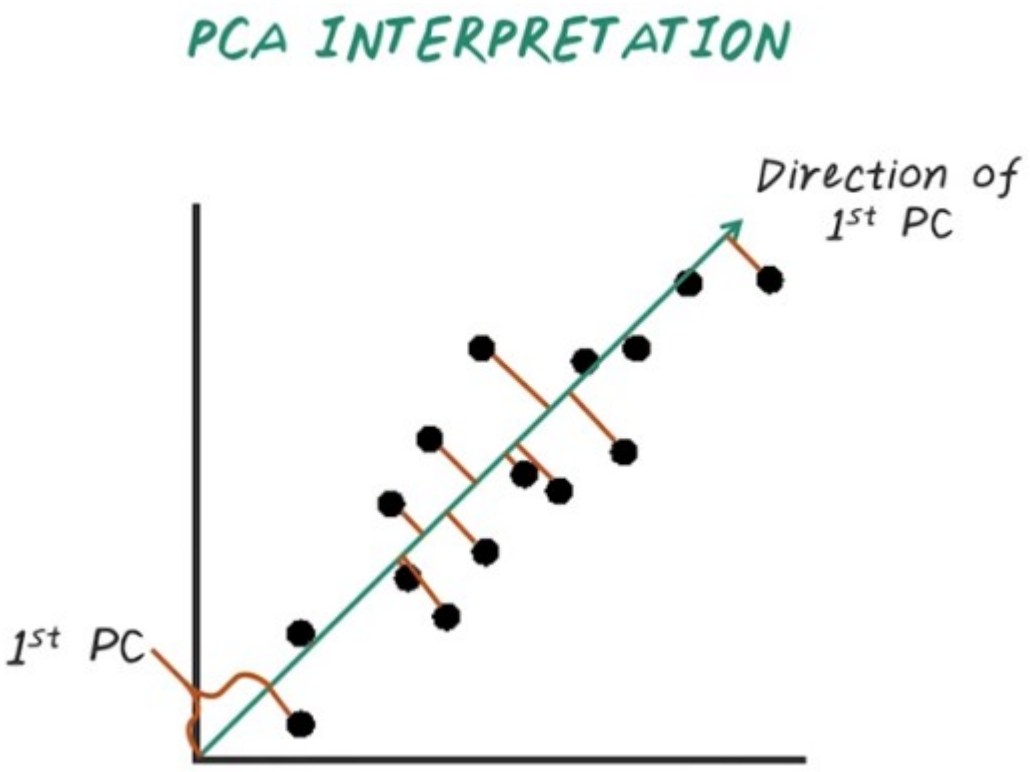


以下 d 為 document 指標, t 為 term 指標
A_dt: A^T A = A_td A_dt' = (...)_tt

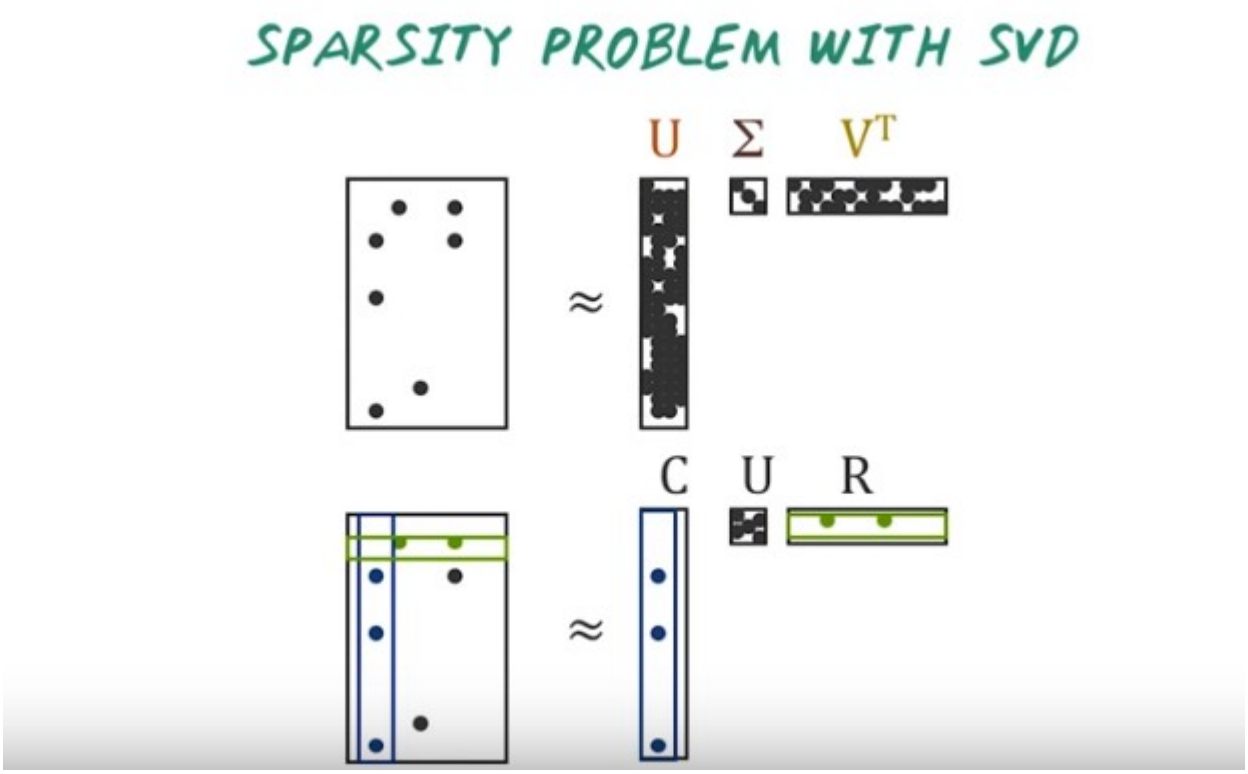
7. The answer is, it's actually a term to term similarity matrix, because when you compute A transpose A, the number of rows and columns of this resulting matrix equals the number of terms. And every element represents the similarity between two pairs of terms.



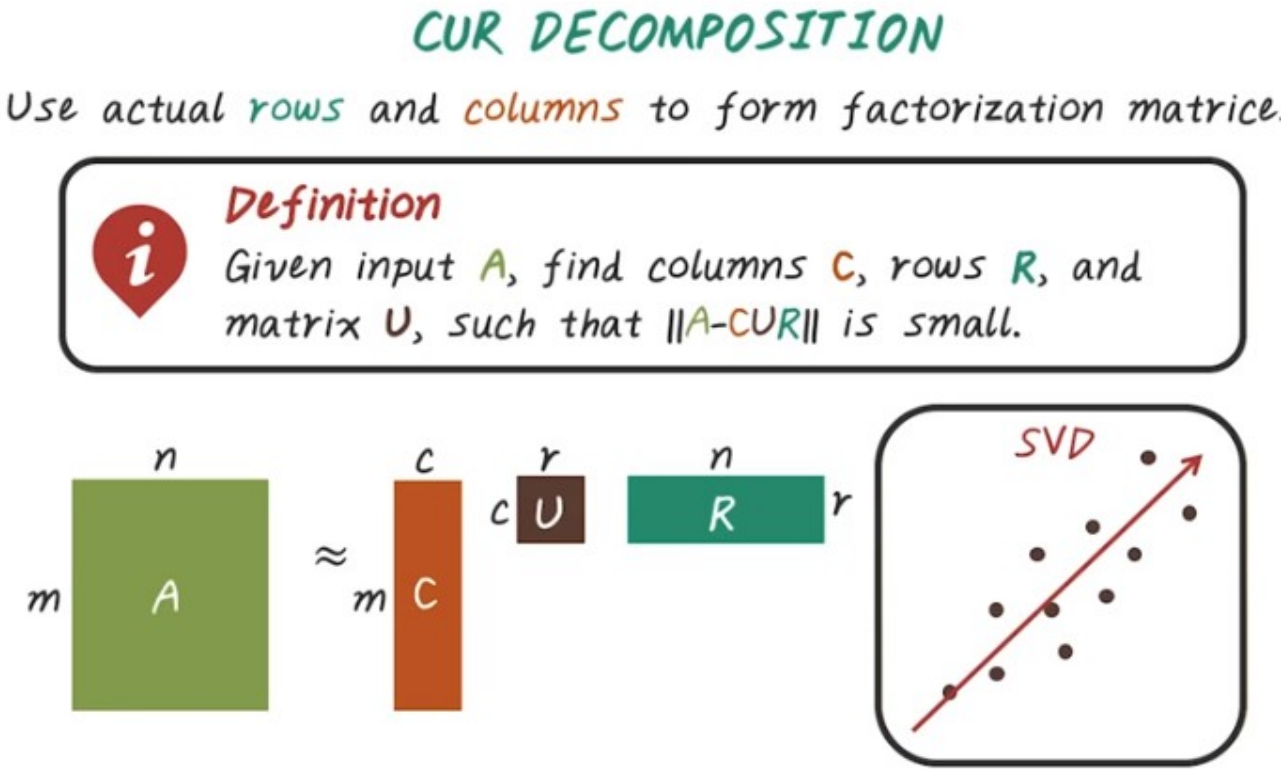
8. Next lets talk about principal component analysis, PCA. PCA is very related to SVD. So we already know, using SVD, we can factorize the matrix x as a product of three matrixes U sigma V transpose. Now if we group U and sigma together into one matrix and we call that principal components, and V transpose, we call that loading. This essentially give us the principal component analysis. So visually, given this large input matrix, n by m, and every row represent a patient, every column represent a disease then we can factorize this into U sigma and V transpose, and if we group U and sigma together and the result of U times sigma become the principal components, and V transpose become the loading matrix. And in particular, this U1 times Sigma1 (上圖方括號中的紅色的) give us the first principle component. The direction of the first principle component is specified by the corresponding vectors in the loading matrix.



9. Now let's illustrate PCA with a visual representation. Given a set of two dimensional points scattered like this. If we want to reduce this set of two dimensional points into one dimensional space, we need to find a direction to project those points to. For example, the first principle component direction is pointing this way, and if we project all those points onto this first principle components, the corresponding offset along this direction will become the coefficient to represent those points. And they are the first principle components. For example, if x axis is represent weight and y axis represent height, every data point represents a specific individual, then if we want to project those individuals into a one dimensional space, and this one dimension is a linear combination of weight and height and the offset on this one dimension is the first principle component.



10. So what's the problem with SVD or PCA? It has a sparsity problem. For example most of the input dataset for analytics are often sparse, meaning that only a small number of elements in the large matrix are non zeros, majority of these elements are zeros. For example, every row represent a patient, every column represent a possible disease, then for a given patient, there's only a few disease this patient has, so this input matrix is very sparse. So SVD is one of the best dimensionality reduction approach, because it gives the best low rank approximation. However the result of SVD, in particular the U and V matrices, are large and dense, so SVD destroys the sparsity in the original data. As a consequence, the result from SVD would take a lot more storage space and competition with time for the subsequent analysis. Alternatively, we may want to use those factorizations that preserves the sparsity. For example, we can use actual columns and actually rows from the original matrix to form the factorization, and this is called CUR decomposition. C represent the sample columns from the original matrix and R represent the sample row from the original matrix. So CUR, in this case, maintains the sparsity of the original data. As a result, the storage cost of CUR decomposition is much smaller than SVD.



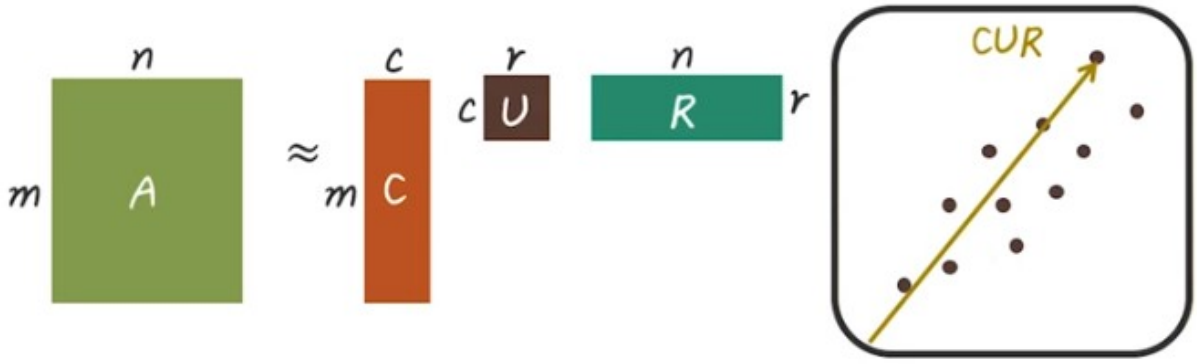
11. So CUR decomposition uses actual rows and columns to form the factorizations. So here's the definition for CUR. Given an input matrix A, find a set of columns in C and a set of row in R and a matrix U, such that the norm of A minus CUR is small. This norm must indicate the errors of approximating our original matrix A using C times U times R. So visually given the input matrix A, which is m by n, we want to approximate A with a smaller matrix C which is m by c, times a small matrix c by r, and a matrix R, which is n by r. Here, C come from the columns of A, and R come from row in A. Let's use this two dimensional example to compare CUR to SVD. So SVD will try to find the best direction to project all those data points to. And this direction often is a linear combination of all these data points.

CUR DECOMPOSITION

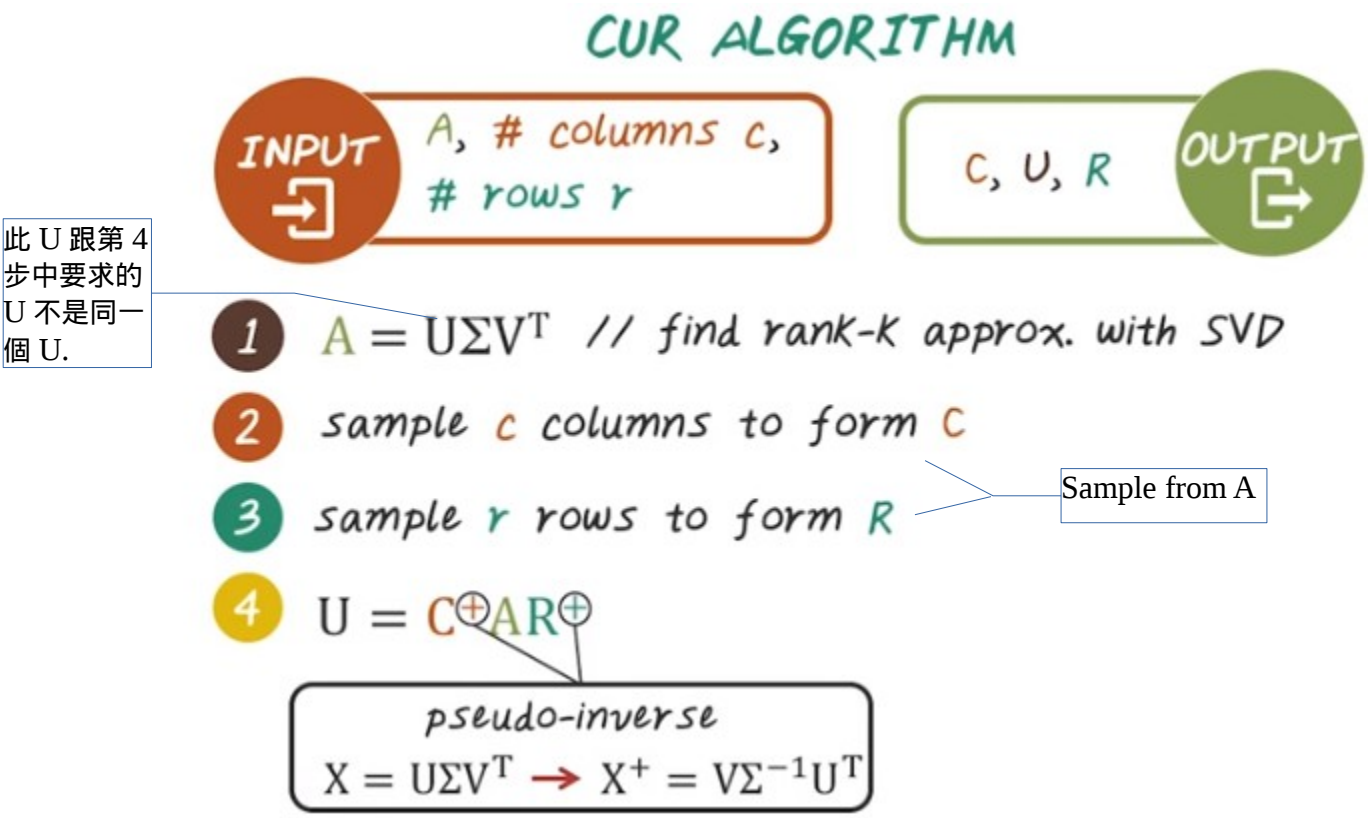
Use actual rows and columns to form factorization matrices.

Definition

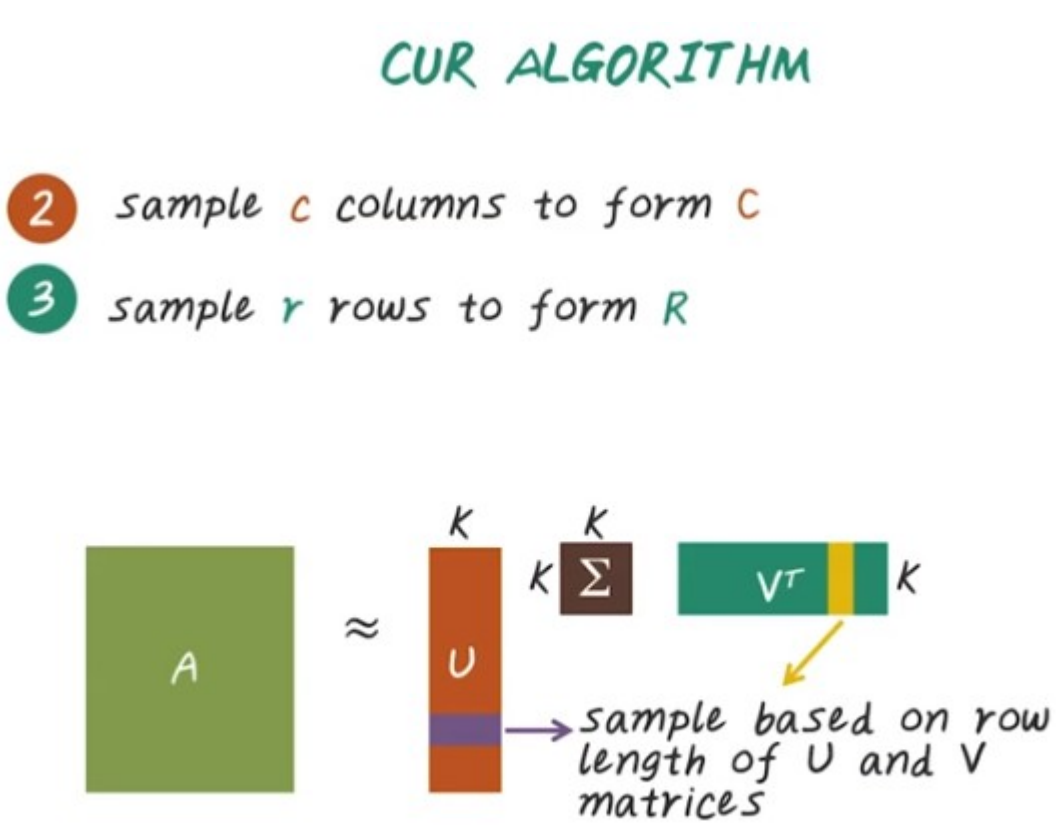
Given input A, find columns C, rows R, and matrix U, such that ||A-CUR|| is small.



However, CUR will try to find an actual data point. And to project the rest of data points towards that direction. For example, CUR may sample this data point over here, then try to project all the other data point to this direction.

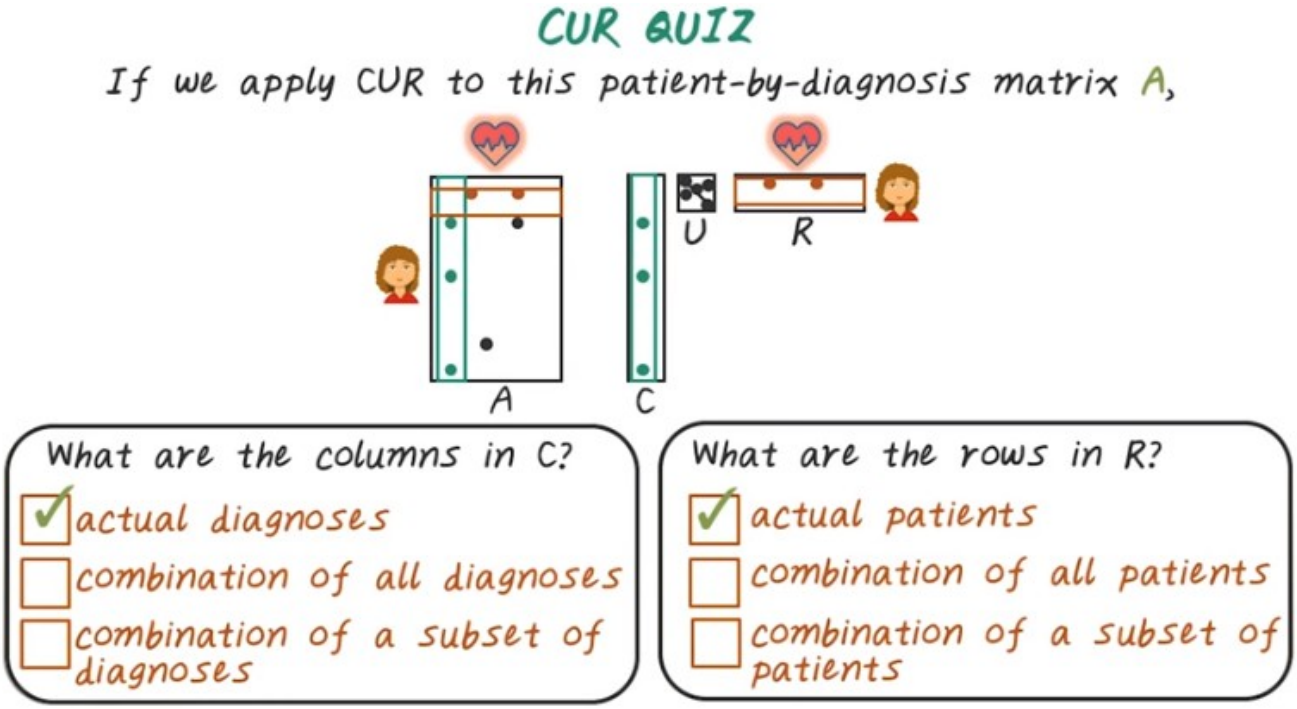


12. Next, let's talk about the actual algorithm for CUR. In the input to CUR algorithms is the matrix A and the number of columns we want to sample, c, and the number of row we want to sample, r. And the output of CUR is these three matrices. C, U, and R. There are many different algorithms that achieve CUR the composition. There's different computational complexity and different approximation error guarantees. And this lecture will show you an algorithm for CUR that based on SV. So, in this specific algorithm, the first step is we do SVD decomposition. Find a top rank-k approximation that's the SVD. Then we sample c columns to form matrix C, and sample R rows to form matrix R. Finally the U matrix can be computed as C pseudo inverse times A times R pseudo inverse. So this symbol represents pseudo-inverse. Pseudo-inverse of matrix X can be computed with SVD. So if we have SVD of X which is U times Sigma times V transpose. And the pseudo-inverse of X is actually V times sigma-inverse, times U Transpose. And the property of pseudo-inverse is X times X-plus, equal to identity.

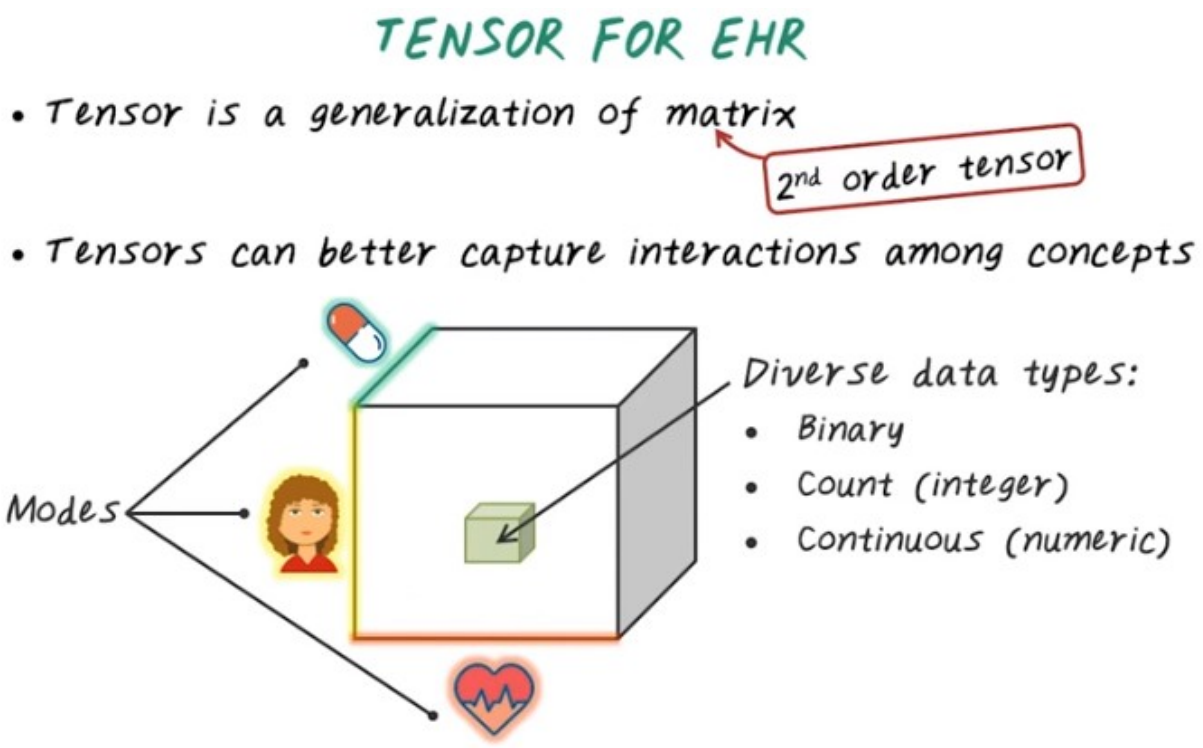


Next, we talk about these two key steps: how do we sample columns and rows based on SVD. So from the previous step, step one, we have this rank-k approximation using SVD. Approximate matrix A by U times sigma times V transpose. Here we have K columns in U and the same for V. Sigma is K by K matrix, were actually sampled based on the row lengths of U and V matrix. If the row length (某一行的 row length 應該是指該行中非 0 元素能扩展出的長度) of this corresponding patient (視頻中指向的 U, 但由下面的 quiz 知, 實際上是 sample from A, 這裡只是定出選 A 中的哪一行) is large, then we're more likely to sample this patient to form R. Similarly, if the row lengths in this V matrix is large, then we're more likely to sample the corresponding columns to form C. So, the probability of sampling is proportional to the row lengths of U and V matrix.

13. Here's a quiz on CUR decomposition. If we apply CUR on this patient-by-diagnosis matrix A, to get the CUR decomposition from this patient-by-diagnosis matrix, then what are the columns in C? Are they actual diagnoses, combination of all diagnoses, or combination of a subset of diagnoses? Similarly, what are the rows in R? Are they actual patients, combination of all patients, or a combination of a subset of patients?

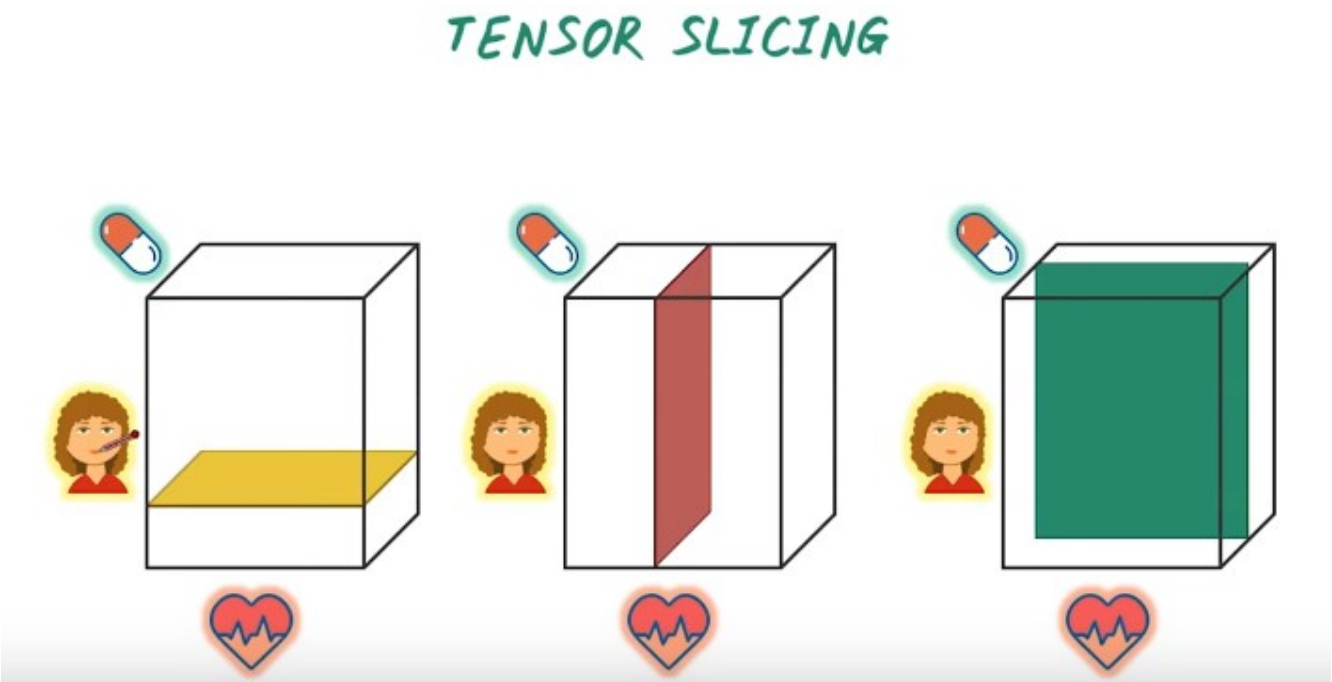


14. And the answers are actual diagnoses and actual patients. So that's a key characteristic of CUR decomposition. You sample actual columns and row from the input matrix. In this case, we'll actually get diagnosis and patient from the original matrix to form the C and R matrices. So they are more intuitive, and also preserves the sparsity in the original data.

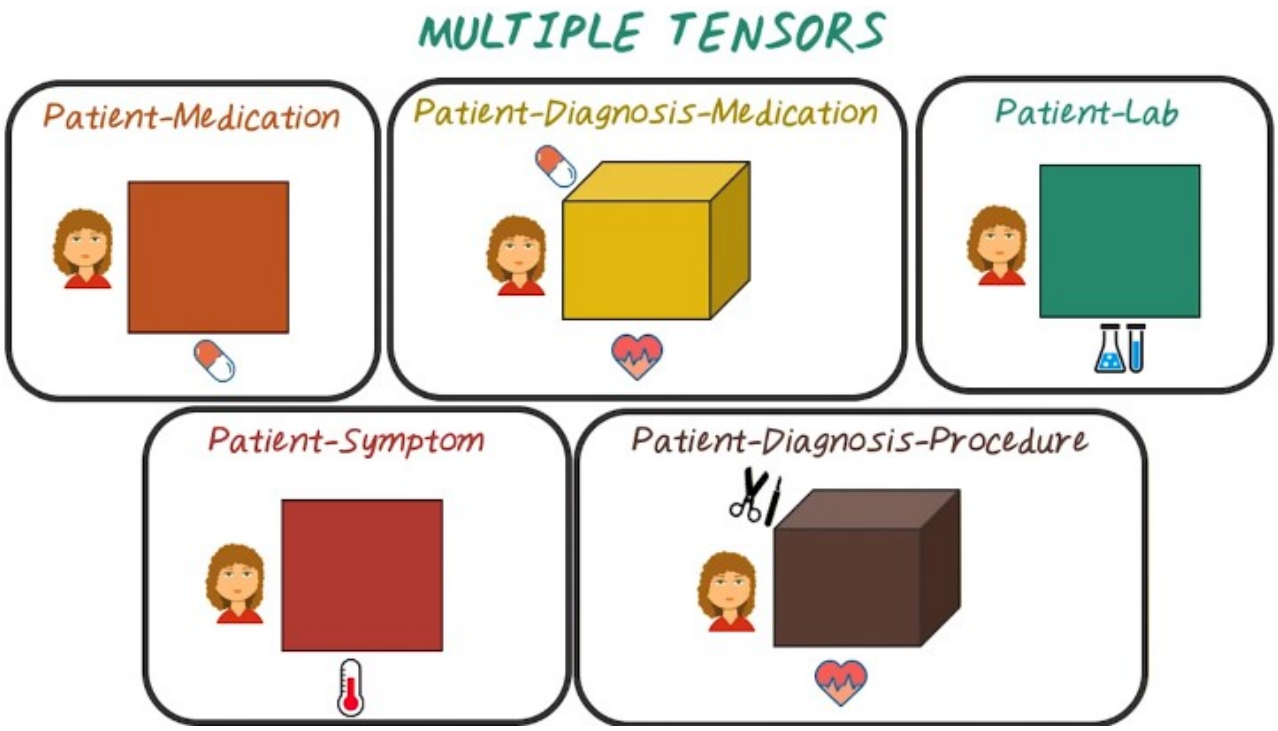


diagnosis: 診斷, 判斷

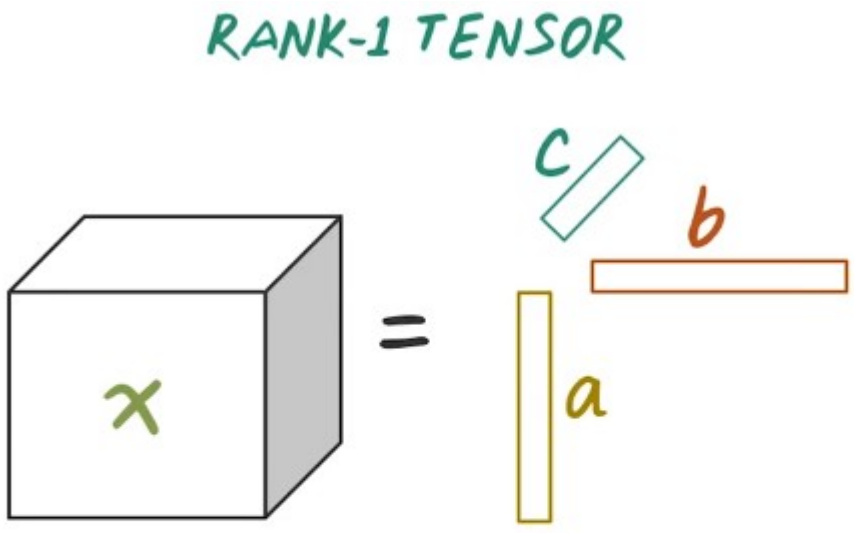
15. Next, we want to talk about tensor factorization as another dimensionality reduction method, but first, what is a tensor? Tensor is a generalization of a matrix. A matrix is actually a special case of tensor of the second order, so we call matrix a second order tensor. Tensor can better capture interactions across concepts. For example, we can have a patient by diagnosis(), by medication tensor, a third order tensor, and we call those 'patient diagnosis medication' or the tensor mode. This element indicate, for this given patient, what diagnosis she has, and for treating this diagnosis, what medication has been given. This tensor representation can capture diverse data types, so the element of this tensor can be quite diverse. It can be binary, indicate whether patient has been taking this drug treating this disease, or it could be count, or integer, at counting the number of times such diagnoses has been given, and for treating this diagnosis, this number of medication has been given. Or it can be some numerical continuous values, so it's quite general.



16. Now let's talk about some basic operation on tensor. Tensor slicing is an operation to extract a subtensor, in this case, a matrix. So the idea is to set all the mode except two modes the same. By doing so, we'll get a matrix. For example, given this third alter tensor, patient, diagnosis, medications, we can get a diagnosis medication matrix for a specific patient. So that's tensor slicing along this particular patient dimension, for example, this healthy patient we can extract a specific matrix for her. Then for a sick patient, we can extract another slice of this tensor that corresponding to this patient that is sick. Similarly, we can extract patients associated with medication for treating a specific disease, say hypertension. We can also extract all the patients and their corresponding disease, by a particular medication, say beta blocker. So these are all different ways for slicing a tensor to get a matrix.



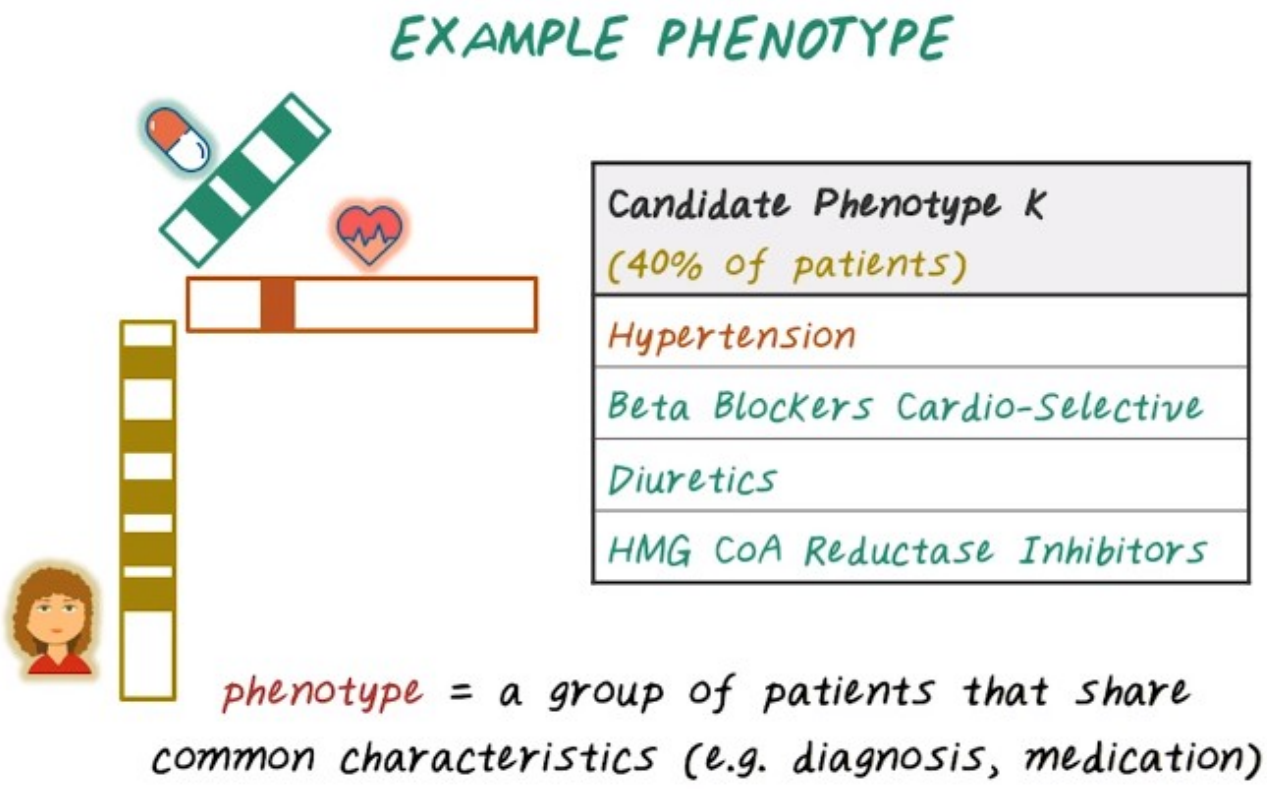
17. We can multiple tensors from electronic health records. For example, we can construct this patient by medication matrix, or second order tensor. We can construct this patient-diagnosis-medication tensor, a third order tensor. We can construct a patient-lab result tensors and we can construct a patient-symptom tensor, and finally, we can construct a patient-diagnosis by procedure tensor. There's many different ways to construct those tensors from electronic health records.



$$x = a \circ b \circ c$$

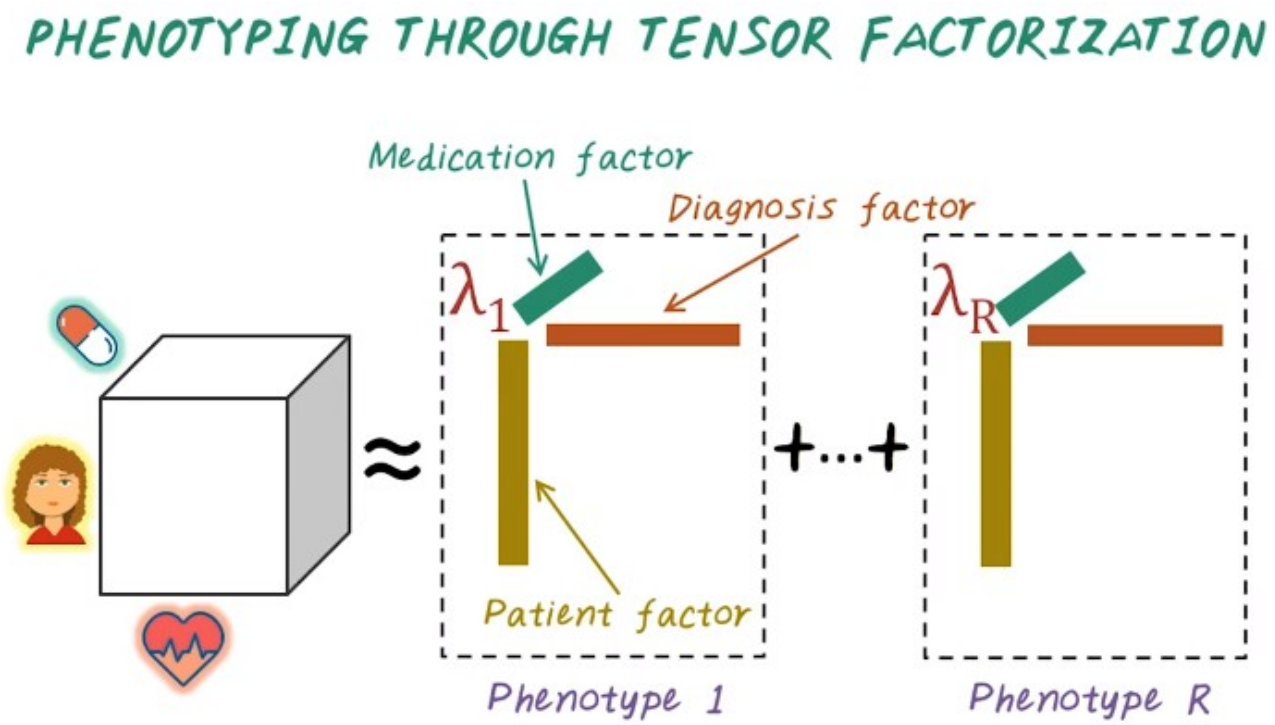
$$x_{ijk} = a_i \cdot b_j \cdot c_k$$

Another important tensor concept is rank-1 tensor. So rank-1 tensor is ultraproduct (就是直積的意思) of a set of vectors, one from each mode. For example, given this third order tensor x, if x is rank-1, then it equals to this other product of three vector, a, b, and c. So the mathematical notation of this rank-1 tensor is represented as this auto product operations of the three vector a, b, c. And the corresponding element in this tensors, this ijk element in this tensors is simply multiplication of the ith element from vector a, the jth element from vector b, and kth element from vector c.

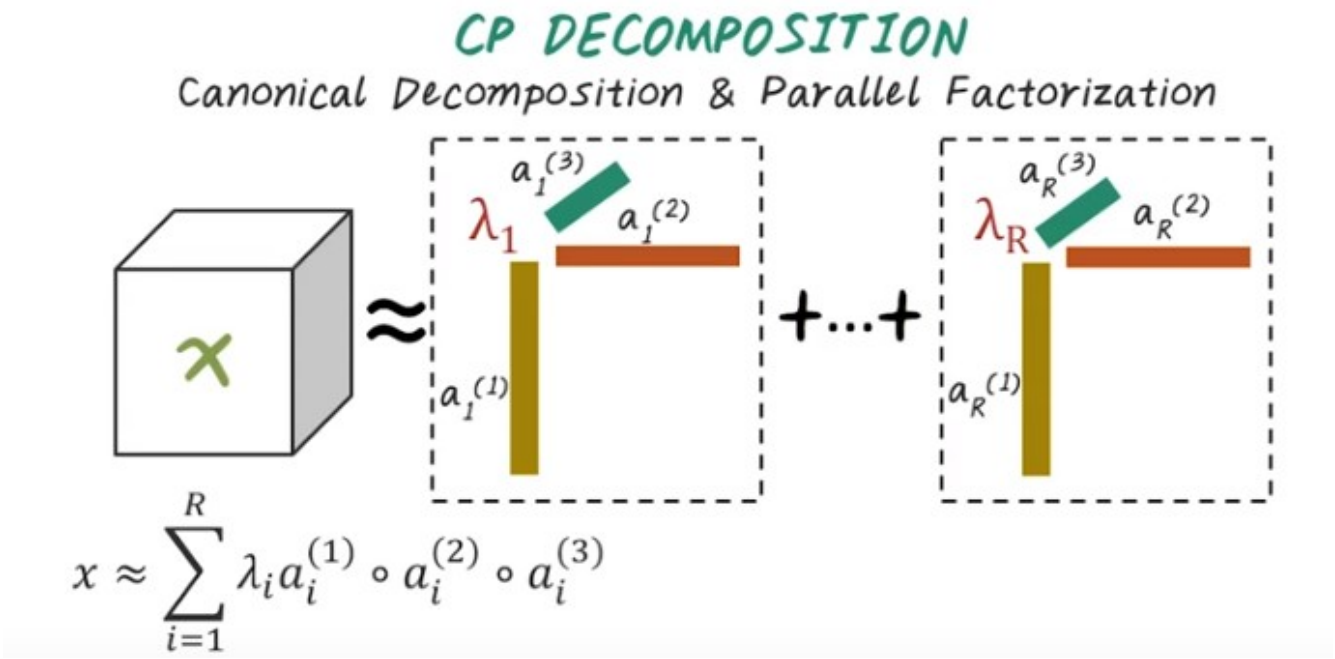


Hypertension: 高血壓

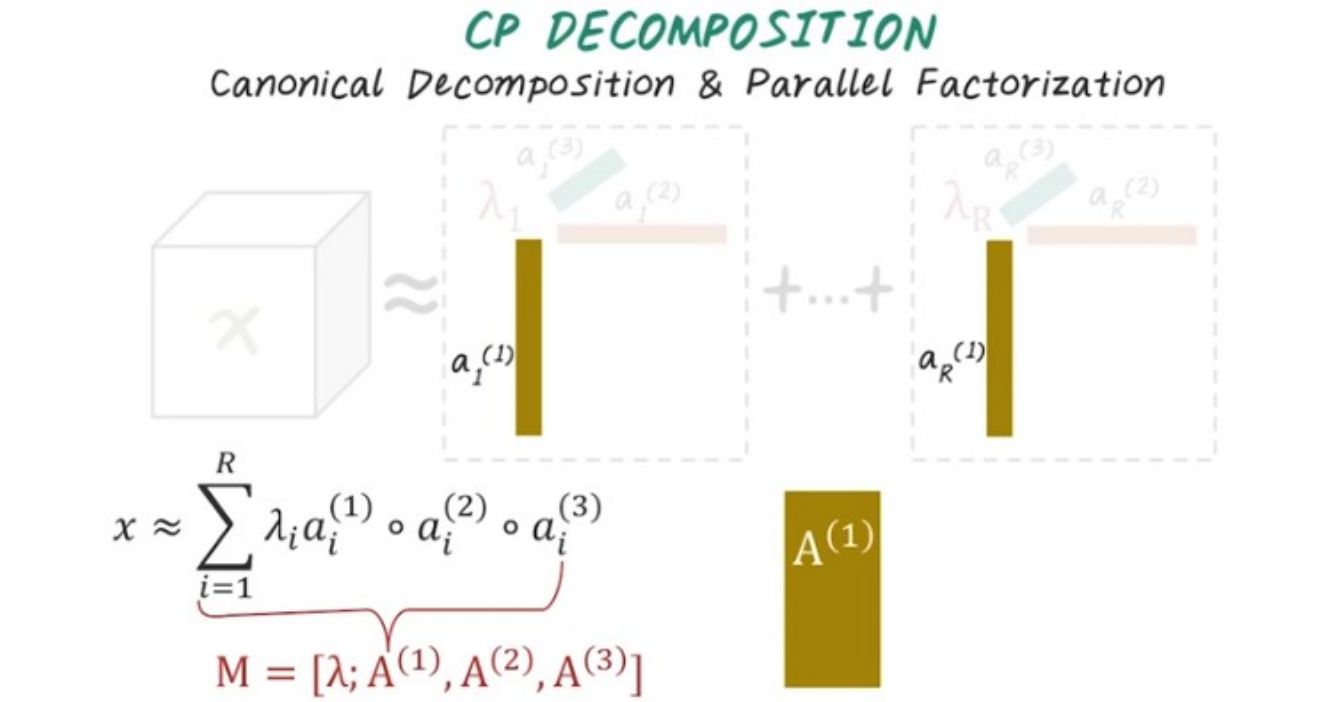
18. Now let's try to connect tensor factorization and rank one tensor to phenotyping. Here's one example of phenotype we want to discover from data. And those phenotypes actually corresponding to a rank one tensor. In this particular case, we have a patient vectors, diagnosis vector, and a medication vector. And this give us a rank one tensor, which correspond to a phenotype. So in this particular phenotypes, 40% of patients has this phenotype, meaning that 40% of entries in this patient vectors are non-zeros. The rest are zeros. The corresponding diagnosis in this diagnosis vectors is hypertension. And for treating hypertension, three medication has been commonly prescribed. Beta blocker, diuretic, and so on. So in this case, phenotypes is a group of patient that share common characteristic. They share some common diagnoses and common medications. Next, we'll show you how to use tensor factorization to discover such phenotypes.



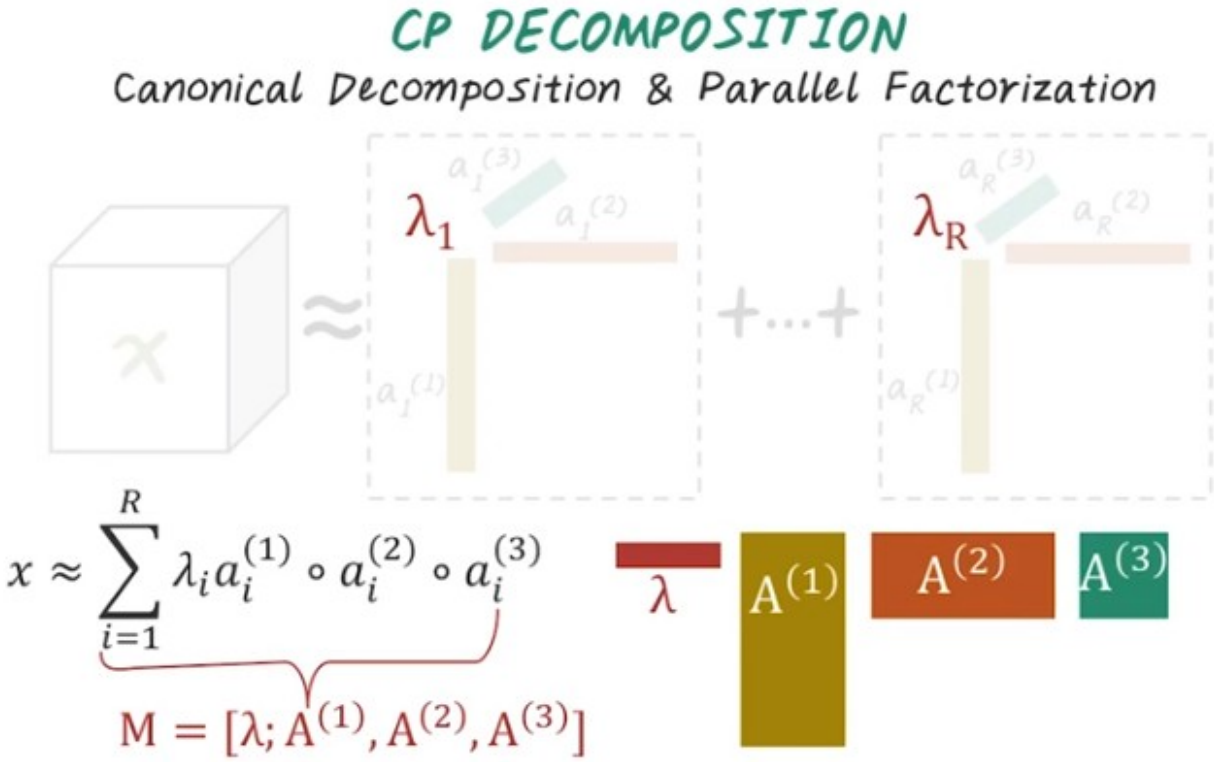
19. Now we want to talk about how do we extract phenotypes using tensor factorization. Given this patient by diagnosis by medication tensor, we can factorize this tensor as the sum of R rank 1 tensor, each one of this rank 1 tensor corresponding to a specific phenotype. Each rank 1 tensors has three factors, patient factors, diagnosis factor, and medication factor. An ultraproduct of this three factors give us a rank 1 approximation of the input tensor, and we have R of those. And combine all of them together, give us approximation of the original tensor, and the lambda corresponding to the importance of these phenotypes. You can imagine different phenotypes may have different importance for representing this input population. So lambda captures that. And this intuitively, how do we use tensor factorization result for phenotyping? Next, let's see what's the computational method for conducting tensor factorization.



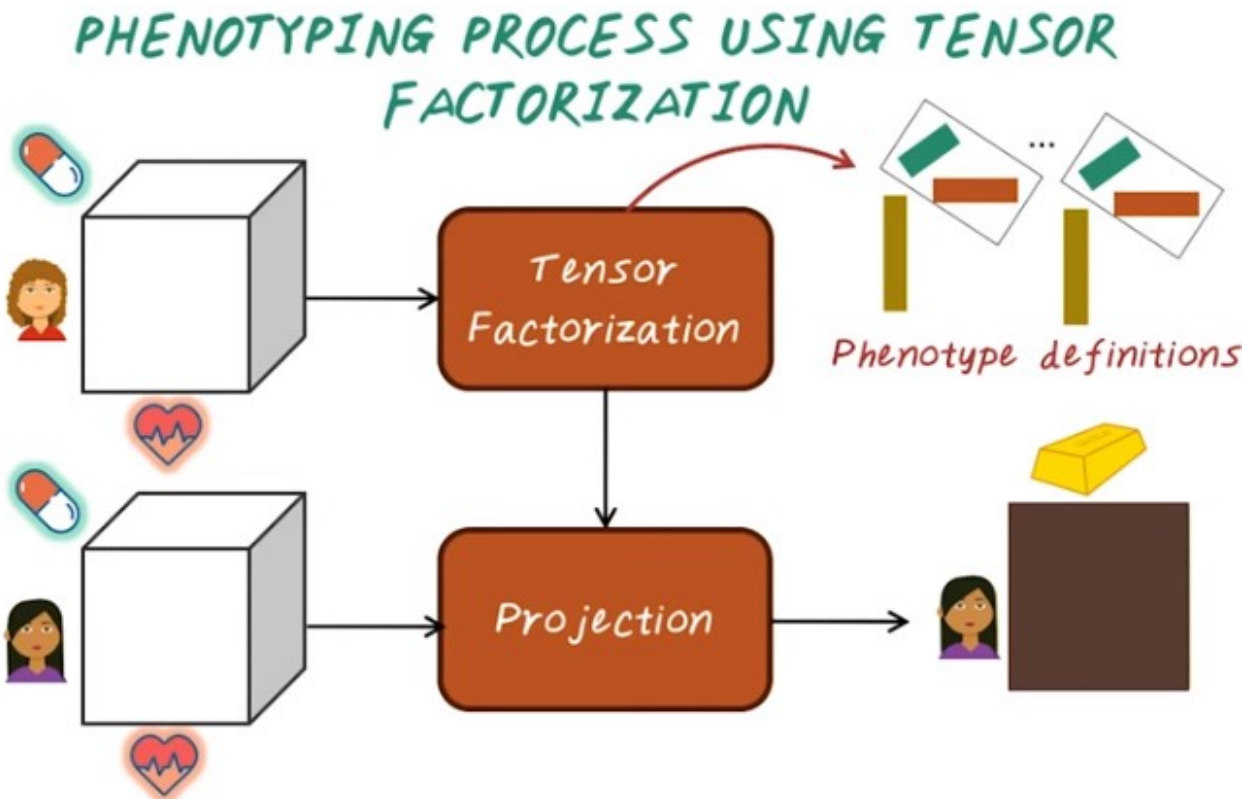
20. CP decomposition factorized input X as sum of a set of rank one testers. Mathematically, it can be represented as tester X, approximated by a sum from one to R and a set of rank one testers. Each rank one tester is represented by a scalar lambda i, and set of vectors, one for each mode, ai1, ai2, and ai3. And this whole thing is the i-th rank one tensor to approximate the input tensor.



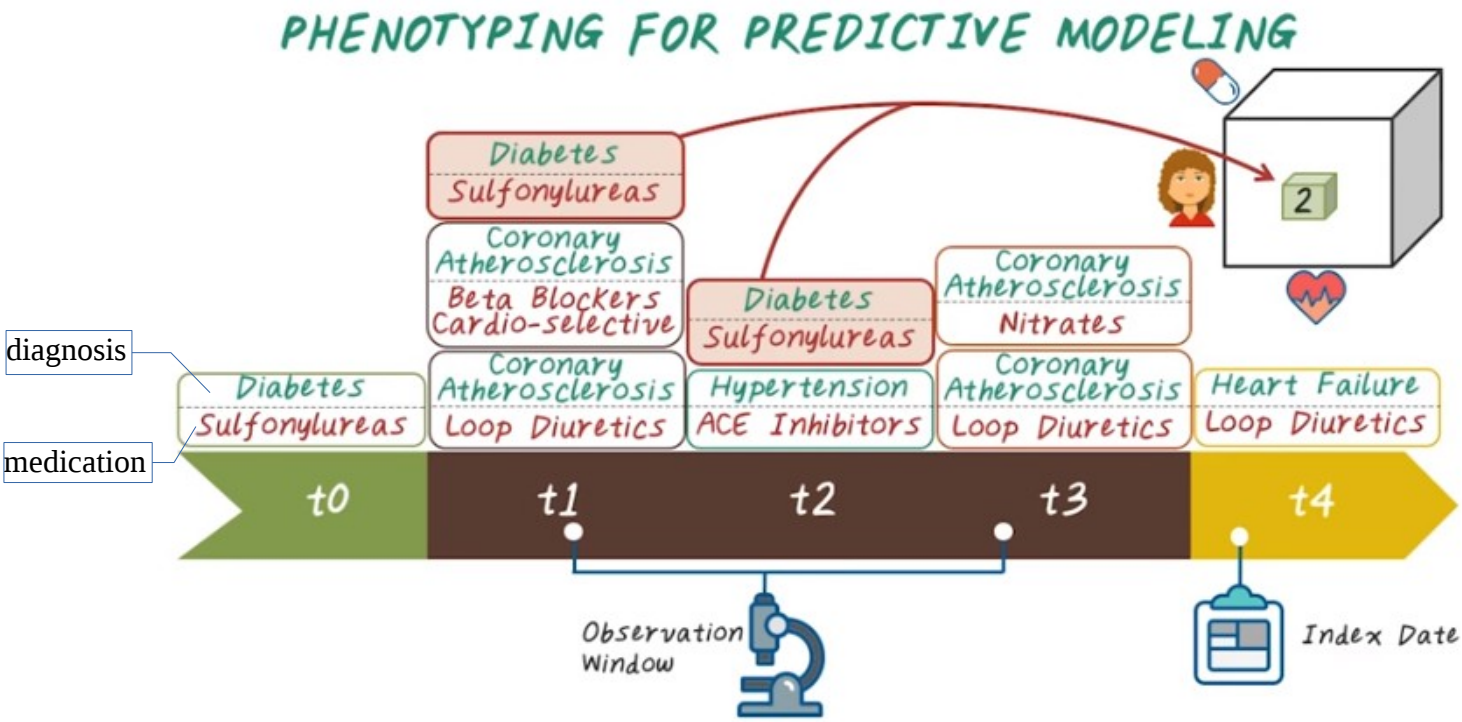
Of course we can reorganize all this vectors into matrices. And that give us the model (視頻中指向的M). So visually, what it does is, all those corresponding vectors, coming from the same mode (就是mode), will be put together, become a matrix A(1).



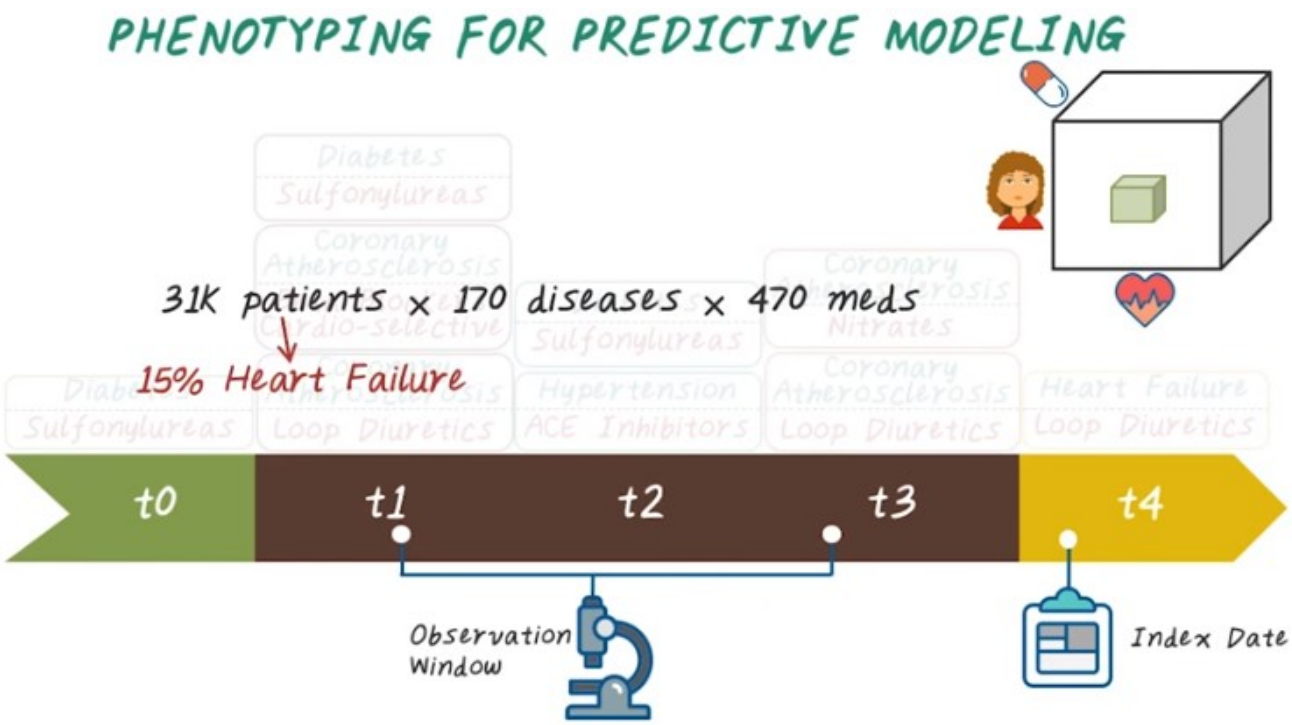
And similarly for all those vectors coming from the second mode, will be put together so we get A(2). And finally we get A(3). All those three matrices are part of the model, and that's the output of CP decomposition. Again, all those corresponding lambdas scalar values tells us how important each rank one tensors are also being put together into this vector lambda. And this whole thing give us the model for CP. And this is the high level intuition of CP decomposition. Like SVD, another matrix factorization, tensor factorization start to become a standard operations. Depending on the loss function and different constraint put onto those components, there is different algorithm to perform CP like decomposition. For phenotyping application we use a variant of CP Decomposition with non-negative constraints. However, overall as a data analyst, you can probably treat this tensor factorization as a black box, just like SVD and PCA.



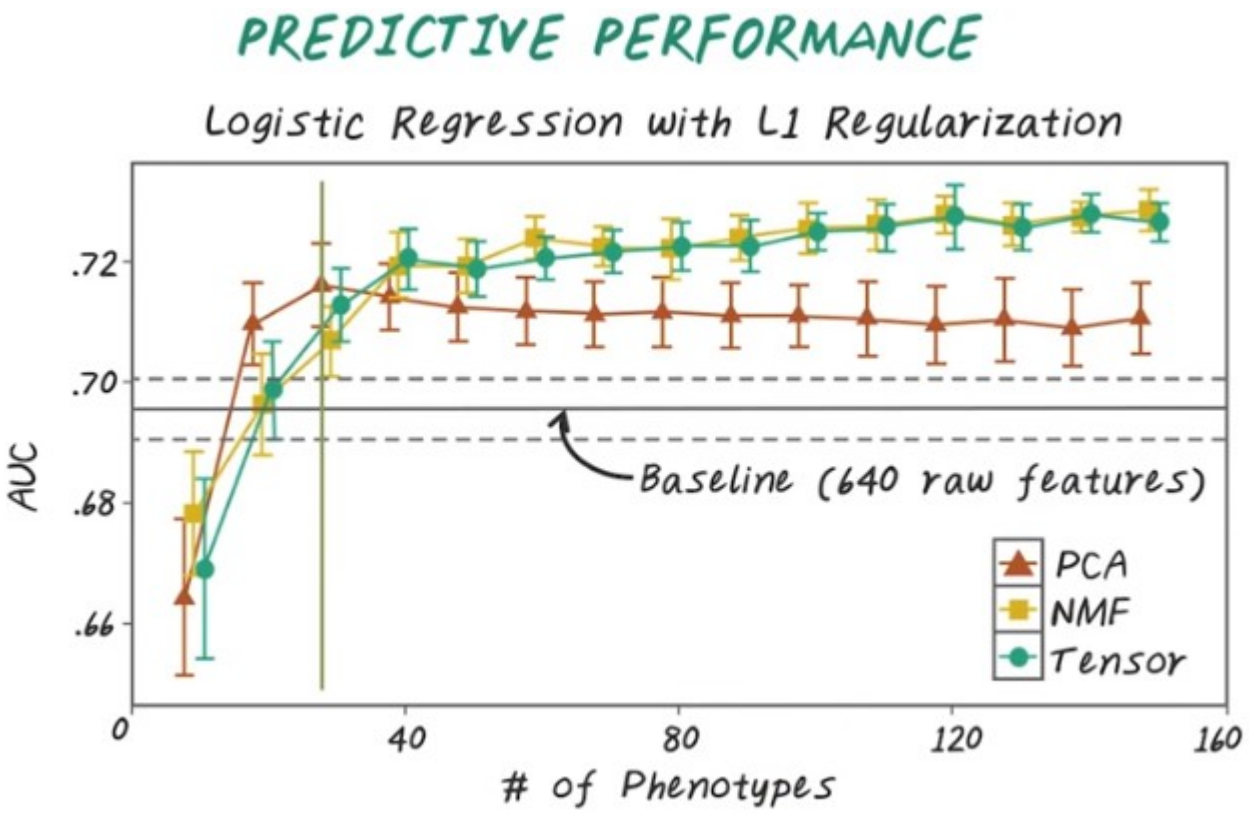
21. So what is the process for using tensor factorization for phenotyping? So you already have input tensors. Using tensor factorization, you have already learned a set of phenotypes. In particular, you have learned the phenotype definition based on the combination of diagnosis and medication. Next a new set of patient comes in. How do you apply those existing phenotype definitions to this new patient? In fact, just like PCA, you can project this new patient's data towards the direction, as specified by the phenotype definition. Then you will get a low dimensional representation, which is every row represent a patient, and every column represent a phenotype. For example, this would tells us, for a given patient, which phenotypes she has.



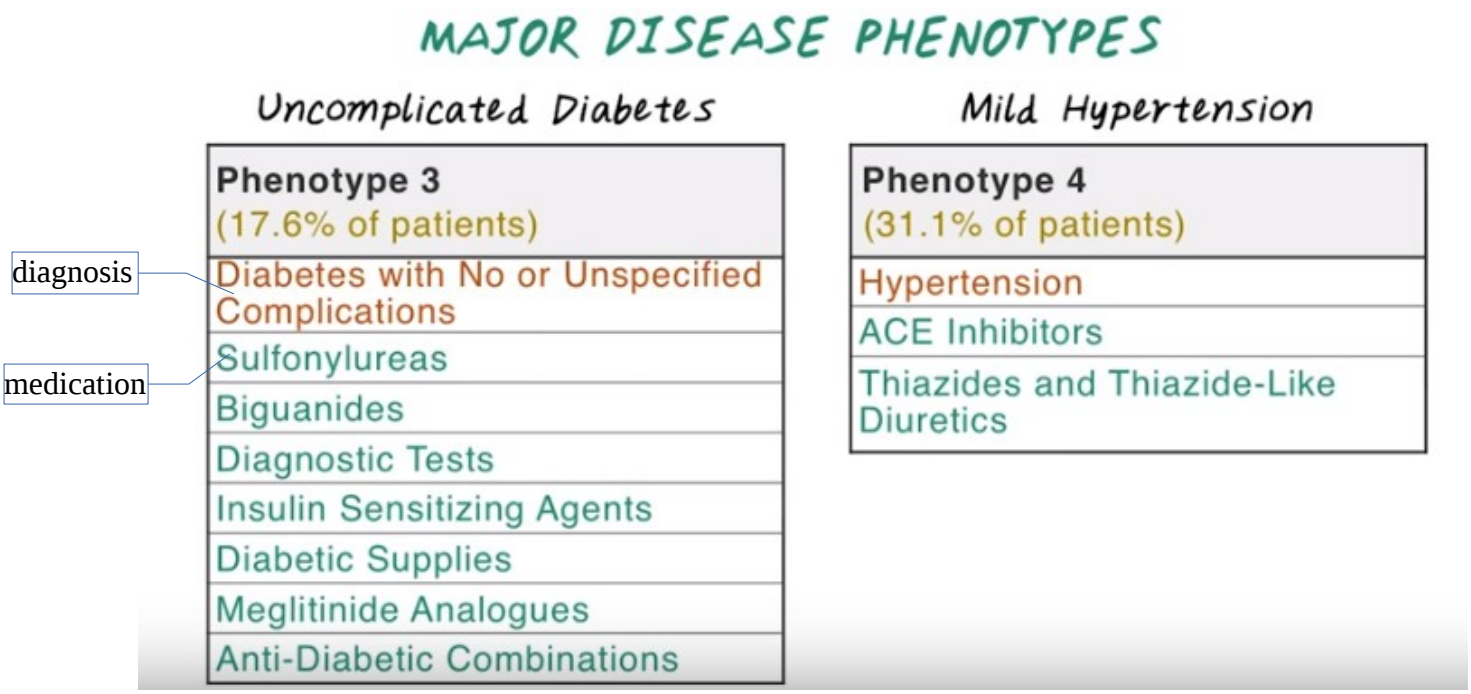
22. Next, let's see how we can construct tensor vectorizations for phenotyping and use it for predictive modelling. So patient EHR data are often represented as event sequences. For a specific patient, we have five records, from t0 to t4. Each record contains one or more diagnosis and medication pairs. For example diabetes and sulfonylureas at t0. Then at t1, three different diagnosis and medication pairs happen in this patient record. And t2, two diagnosis and medication pairs happen. And t3, another two pairs. And t4, heart failure happened and loop diuretic has been used. And this is the event sequence for a given patient. And note that the goal of this predictive modeling is to predict heart failure, which is event happen at this time, t4. To construct tensor, we need to find the index day and observation window, then aggregate all this diagnosis medication pairs within the observation window to populate this tensor. For example this patient diagnosed with heart failure at t4, which is the index date. Then we look back two years to construct observation window, then we count the number of occurrences of all diagnosis and medication pairs. For example, this Diabetes and Sulfonylureas happened twice within the observation window. The corresponding element in this tensure will be 2.



Then we go through all the patient to populate the entire tensors, we'll have this patient by diagnosis by medication tensor. Based on the information from the observation window prior to the index date. For example, in this specific case, we can construct a tensor of size 31,000 patients by 170 disease diagnosis by 470 medications. And 15% of patients in this tensor had heart failure. And we want to apply tensor factorization on this tensor to extract phenotypes, then use those phenotypes as features to predict whether patient will have heart failure or not.



23. So here's a predictive performance. Here, we're trying to compare different dimensionality reduction method. And using those low dimensional representation as features. To predict whether the patient will have heart failure or not. In this case, the baseline performance is indicated by this line. Is AUC close to 0.7? Then we're comparing three different methods. a tensor vectorization, non-active matrix vectorization and principle component analysis. And this is the baseline performance using all the raw data, 640 features. And the classifier we're using, are all the same, it's logistic regression with L1 regularization. Then, as we increase the number of phenotypes or increase the load emission of representation. You can see the performance improves for PCA. Similar trend has been observed for non negative matrix factorization, NMF. And also for the tensor method, which is based on non-aggregative tensor factorization. And here, you notice that, with only a small number of phenotypes, in this case, 30. All those dimensionality reduction methods outperforms the baseline method, which use 640 features. This really shows all those dimensionality reduction method really works. So, as we increase the number of phenotypes, you can see tensor method and NMF perform better than PCA. But between those two, they are quite similar in term of predictive performance.



24. Next, I want to show you intuitively how those phenotypes look like. Using tensor factorization, we're able to extract different disease phenotypes, some corresponding to major disease such as diabetes without complications, so we call this uncomplicated diabetes phenotypes that cover 17.6% of of the population. Then we have another phenotypes corresponding to mild hypertension which covers 31.1% of patient that has hypertension, and two other medications commonly used for treating hypertension. In this particular example, the results from the tensor factorization method are presented to a cardiologist, and he thinks those result make sense, and assign label as tags.

DISEASE SUBTYPES

Mild Hypertension	Moderate Hypertension	Severe Hypertension
Phenotype 4 (31.1% of patients)	**Phenotype 2** (31.5% of patients)	**Phenotype 6** (24.3% of patients)
Hypertension	Hypertension	Hypertension
ACE Inhibitors	Beta Blockers Cardio-Selective	Calcium Channel Blockers
Thiazides and Thiazide-Like Diuretics	Angiotensin II Receptor Antagonists	Antihypertensive Combinations
	Loop Diuretics	Antiadrenergic Antihypertensives
	Potassium	Potassium Sparing Diuretics
	Nitrates	
	Alpha-Beta Blockers	
	Vasodilators	

And the tensor factorization method not only can discover major disease phenotypes, but also can discover disease subtypes. Here are three different phenotypes discovered by tensor factorization. They all shared a common diagnosis, which is hypertension. But the medication for treating those patients with hypertension are very different. Because of that, the cardiologists give the labels of these three phenotypes as mild hypertension, moderate hypertension, and severe hypertension.

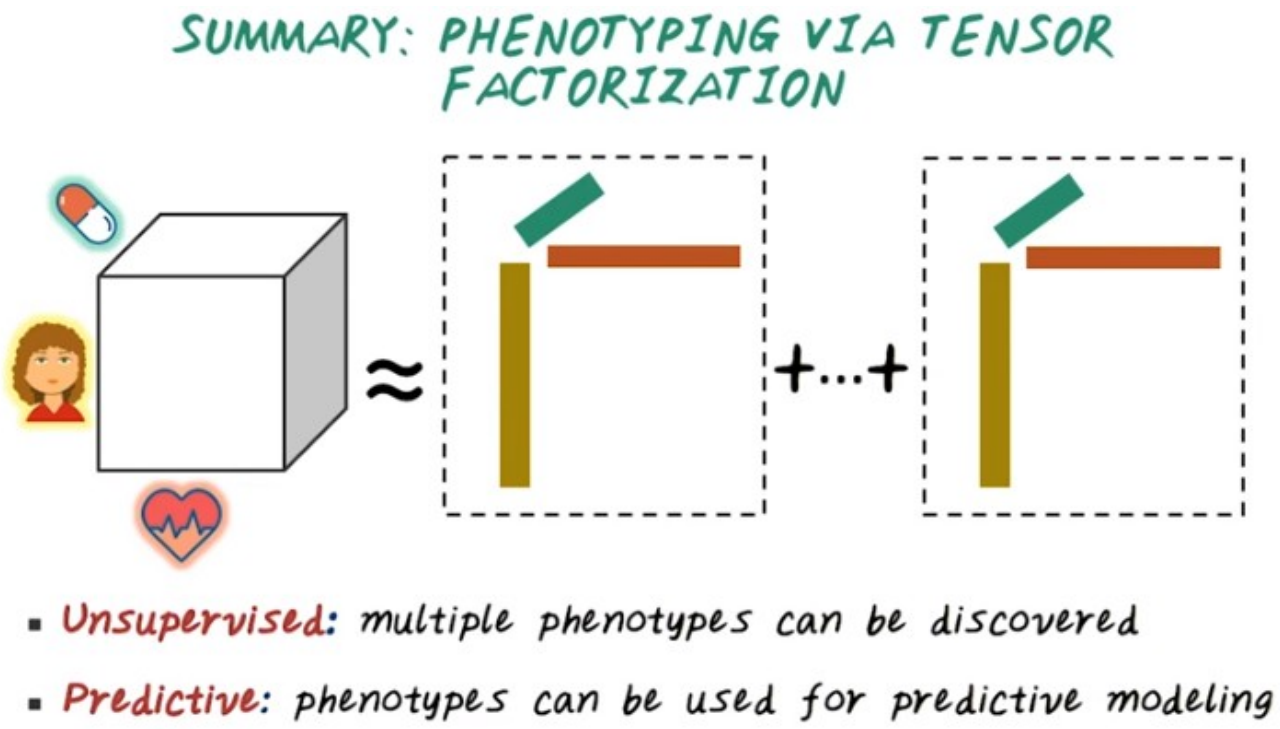
TENSOR VS. NMF

Tensor Phenotype	
Hypertension	0.94
Hypertensive Heart Disease	0.06
Beta Blockers Cardio-Selective	0.51
Calcium Channel Blockers	0.32
Diuretic Combinations	0.06
Nitrates	0.06
HMG CoA Reductase Inhibitors	0.06
Vasodilators	0.05

More concise

NMF Phenotype	
Hypertension - Sympathomimetics	0.0032
Hypertension - Insulin	0.0027
Hypertension - Potassium	0.0018
Hypertension- Beta Blockers Cardio-Selective	0.0004
Hypertension- HMG CoA Reductase Inhibitors	0.0003
Major Symptoms, Abnormalities – Sympathomimetics	0.0167
Major Symptoms, Abnormalities - Insulin	0.0143
Major Symptoms, Abnormalities - Sodium	0.0133
Major Symptoms, Abnormalities - Potassium	0.0097
Major Symptoms, Abnormalities – Coumarin Anticoagulants	0.0092
Vascular Disease - Sympathomimetics	0.0068
Other Gastrointestinal Disorders - Sympathomimetics	0.0065
Other Endocrine/Metabolic/Nutritional Disorders - Sympathomimetics	0.0062
...1,549 total combinations	

25. So you may wonder how does tensor factorization compare to this non negative matrix factorization? So tensor factorization can provide much concise phenotype representation. For example, here's one phenotype coming out of tensor factorization. It consists of two diagnosis and a set of medications. Well for the corresponding phenotypes, in NMF, it looks a lot more complicated because it captured all interaction between diagnosis and medication and the list is much longer. In fact, there is over 1000 different combination of these medication has to be specified in order to summarize these phenotypes. So that the tensor phenotypes is much more concise. As a result, it's much more intuitive to present a tensor phenotype to clinicians.



26. So in summary, we talk about tensor factorization as a way for doing phenotypic, and it represents a patient as the tensor. For example, this patient diagnosis medication tensor, then summarize that tensor as a set of rank one tensors. There's several different benefits for using tensor factorization for phenotyping. First, In the unsupervised method, we can discover multiple phenotypes simultaneously, without any supervision from experts. And the resulting phenotypes can have predictive power. As we have shown you, using those phenotypes, we can predict heart failure better than using the raw data.