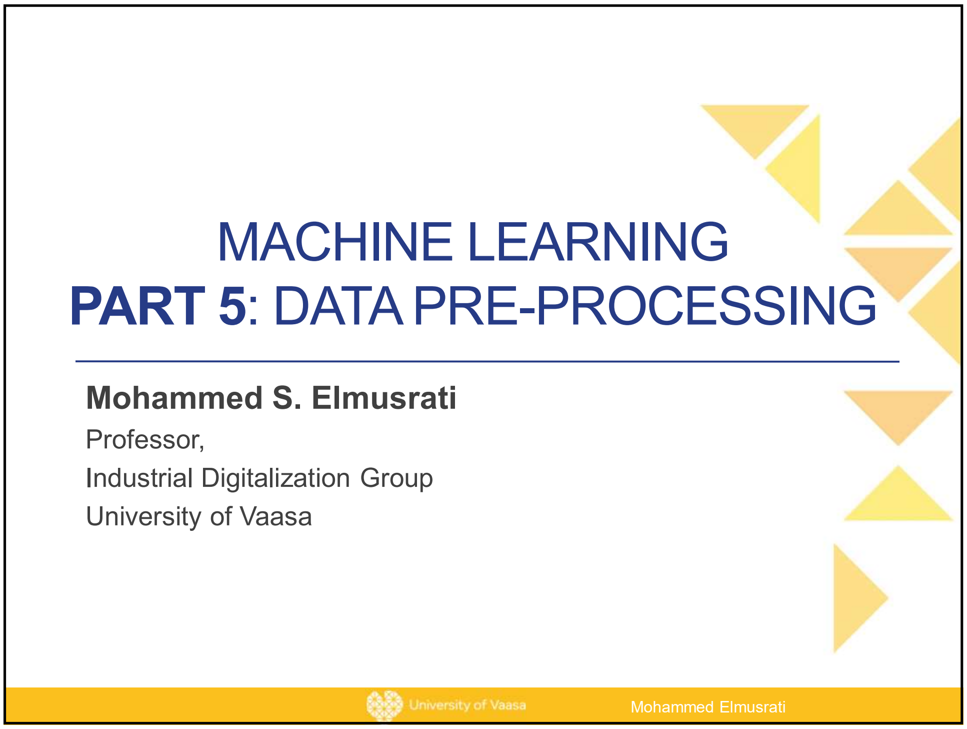


MACHINE LEARNING
PART 5: DATA PRE-PROCESSING

Mohammed S. Elmusrati
Professor,
Industrial Digitalization Group
University of Vaasa

1

Data Quality is Essential for Good Machine Learning Algorithms

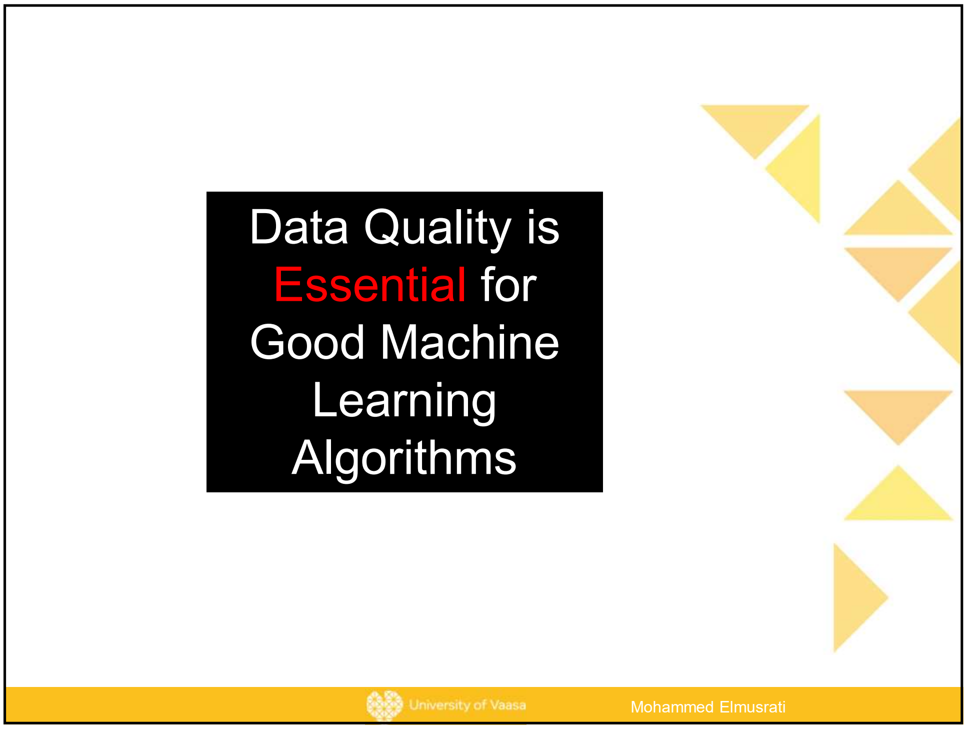
2

Data Quality is Essential for Good Machine Learning Algorithms

University of Vaasa

Mohammed Elmusrati

3

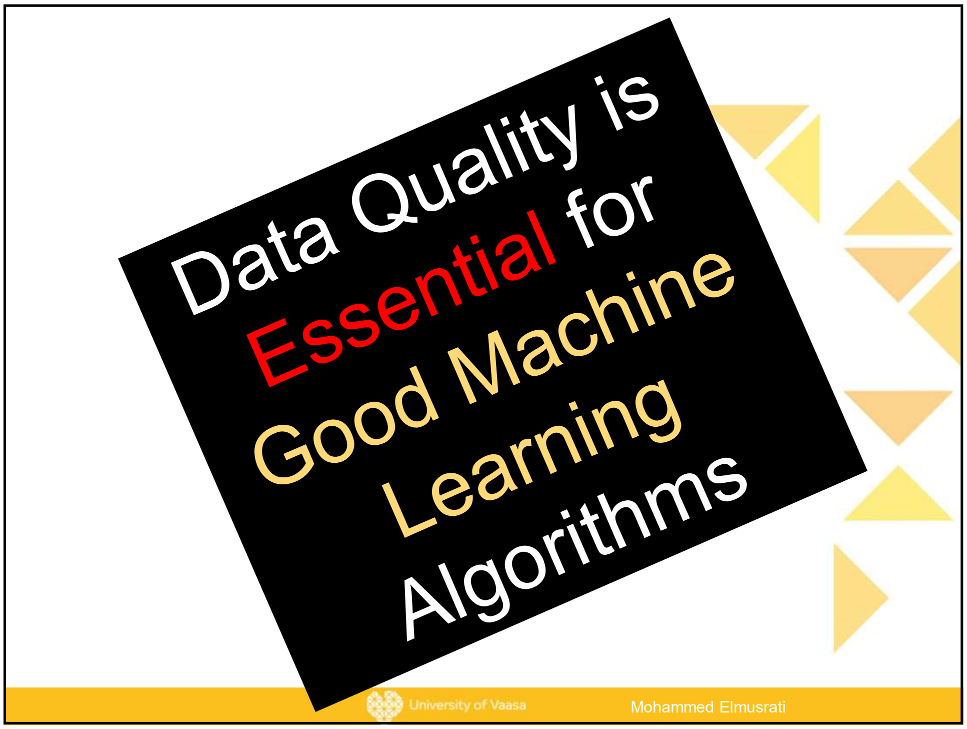


Data Quality is Essential for Good Machine Learning Algorithms

University of Vaasa

Mohammed Elmusrati

4



Data Quality is Essential for Good Machine Learning Algorithms

University of Vaasa Mohammed Elmusrati

5

Data Quality is Essential for Good Machine Learning Algorithms

Data Quality is Essential for Good Machine Learning Algorithms

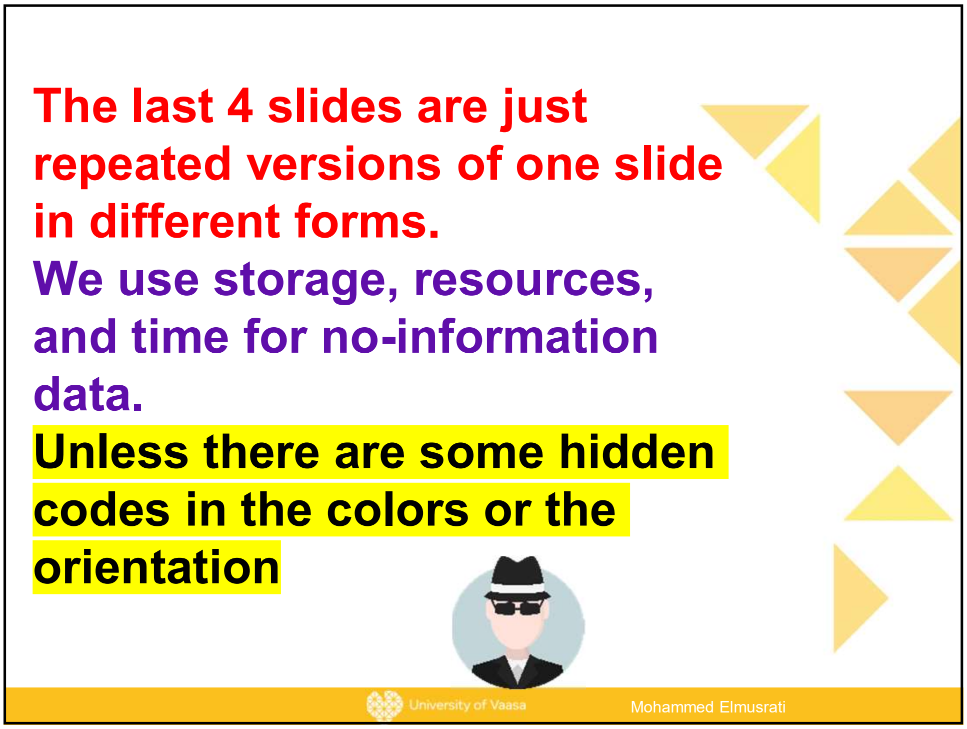
Data Quality is Essential for Good Machine Learning Algorithms

Data Quality is Essential for Good Machine Learning Algorithms

Data Quality is Essential for Good Machine Learning Algorithms

University of Vaasa Mohammed Elmusrati

6



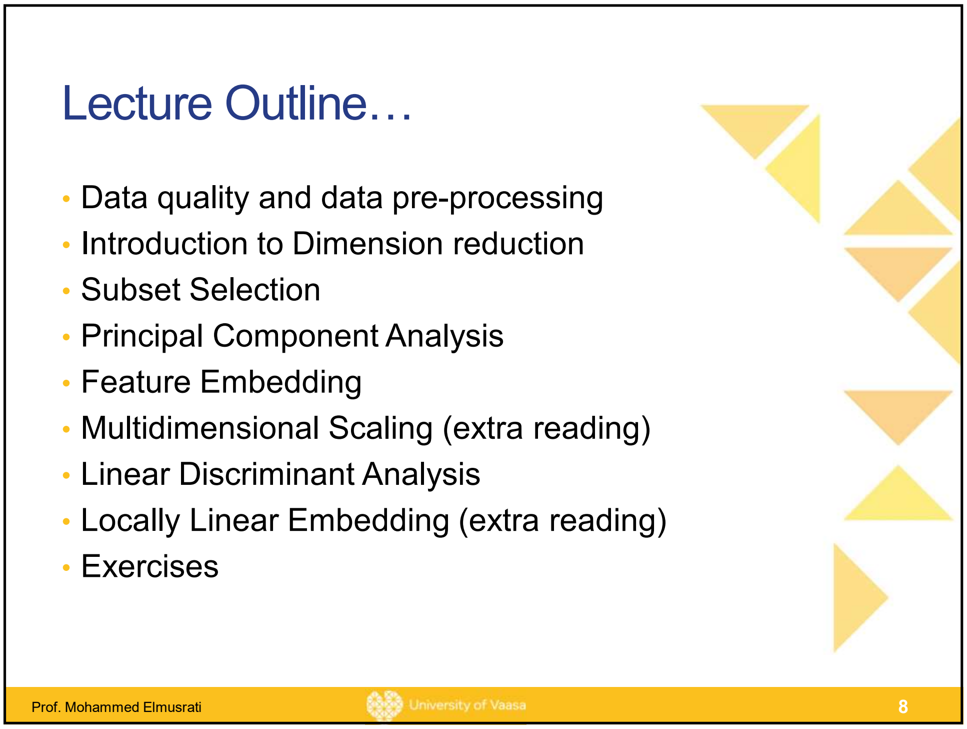
The last 4 slides are just repeated versions of one slide in different forms.

We use storage, resources, and time for no-information data.

Unless there are some hidden codes in the colors or the orientation

University of Vaasa Mohammed Elmusrati

7

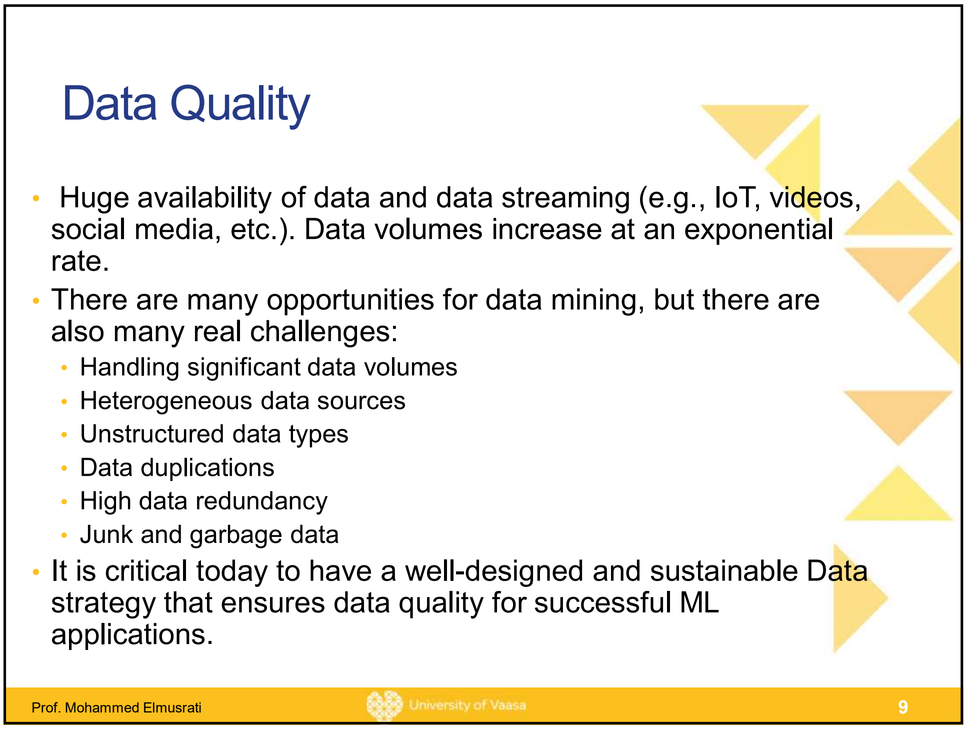


Lecture Outline…

- Data quality and data pre-processing
- Introduction to Dimension reduction
- Subset Selection
- Principal Component Analysis
- Feature Embedding
- Multidimensional Scaling (extra reading)
- Linear Discriminant Analysis
- Locally Linear Embedding (extra reading)
- Exercises

Prof. Mohammed Elmusrati University of Vaasa 8

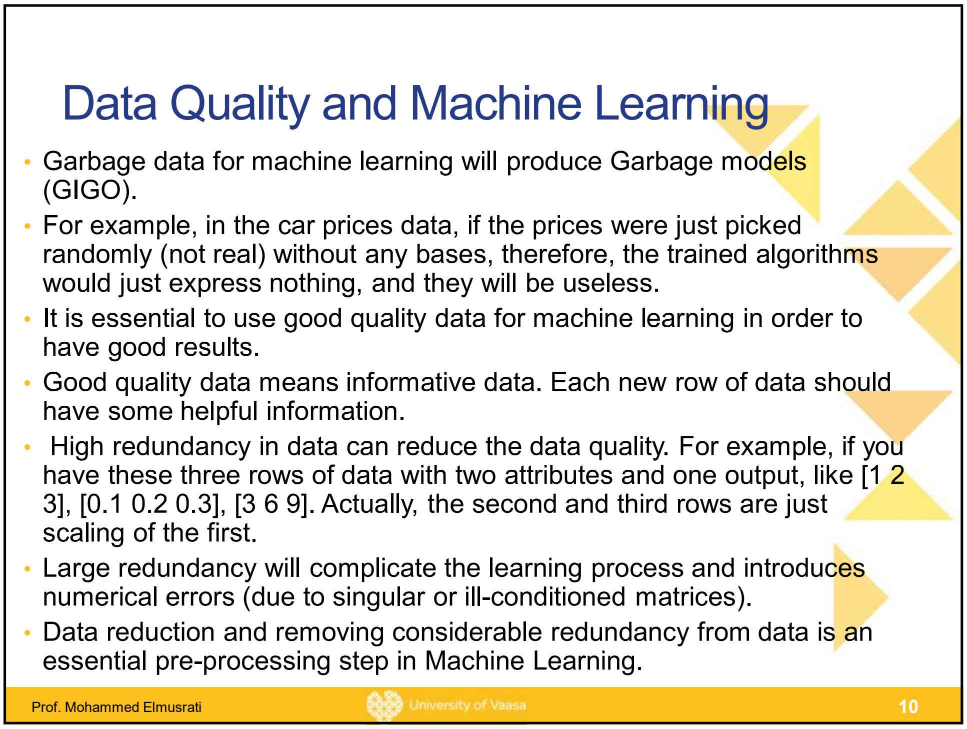
8



Data Quality

- Huge availability of data and data streaming (e.g., IoT, videos, social media, etc.). Data volumes increase at an exponential rate.
- There are many opportunities for data mining, but there are also many real challenges:
  - Handling significant data volumes
  - Heterogeneous data sources
  - Unstructured data types
  - Data duplications
  - High data redundancy
  - Junk and garbage data
- It is critical today to have a well-designed and sustainable Data strategy that ensures data quality for successful ML applications.

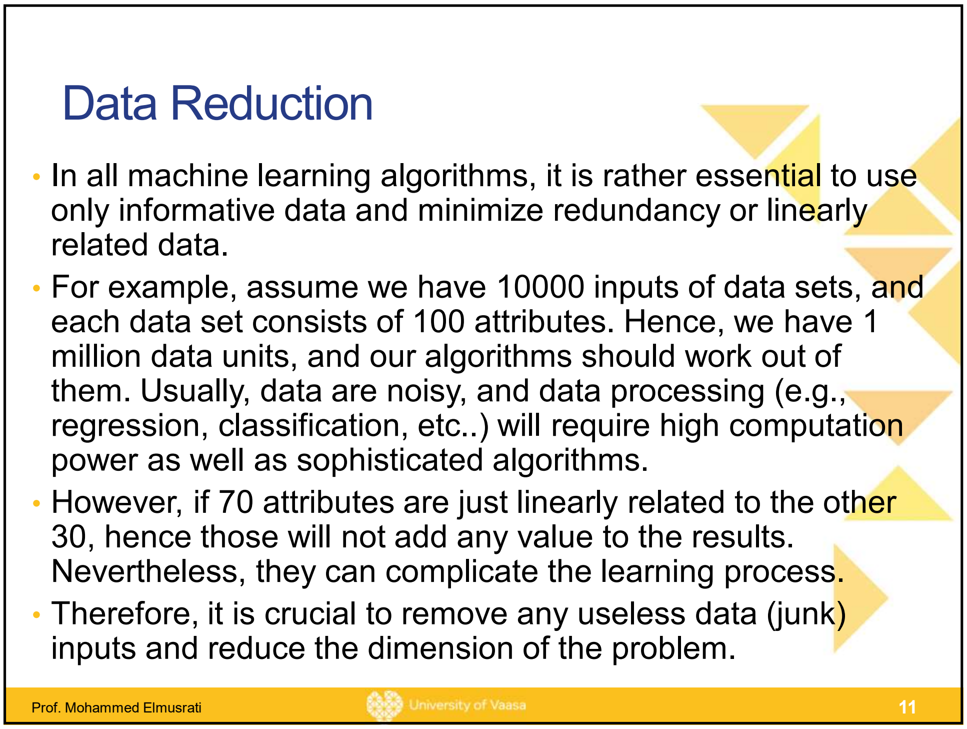
9



Data Quality and Machine Learning

- Garbage data for machine learning will produce Garbage models (GIGO).
- For example, in the car prices data, if the prices were just picked randomly (not real) without any bases, therefore, the trained algorithms would just express nothing, and they will be useless.
- It is essential to use good quality data for machine learning in order to have good results.
- Good quality data means informative data. Each new row of data should have some helpful information.
- High redundancy in data can reduce the data quality. For example, if you have these three rows of data with two attributes and one output, like [1 2 3], [0.1 0.2 0.3], [3 6 9]. Actually, the second and third rows are just scaling of the first.
- Large redundancy will complicate the learning process and introduces numerical errors (due to singular or ill-conditioned matrices).
- Data reduction and removing considerable redundancy from data is an essential pre-processing step in Machine Learning.

10

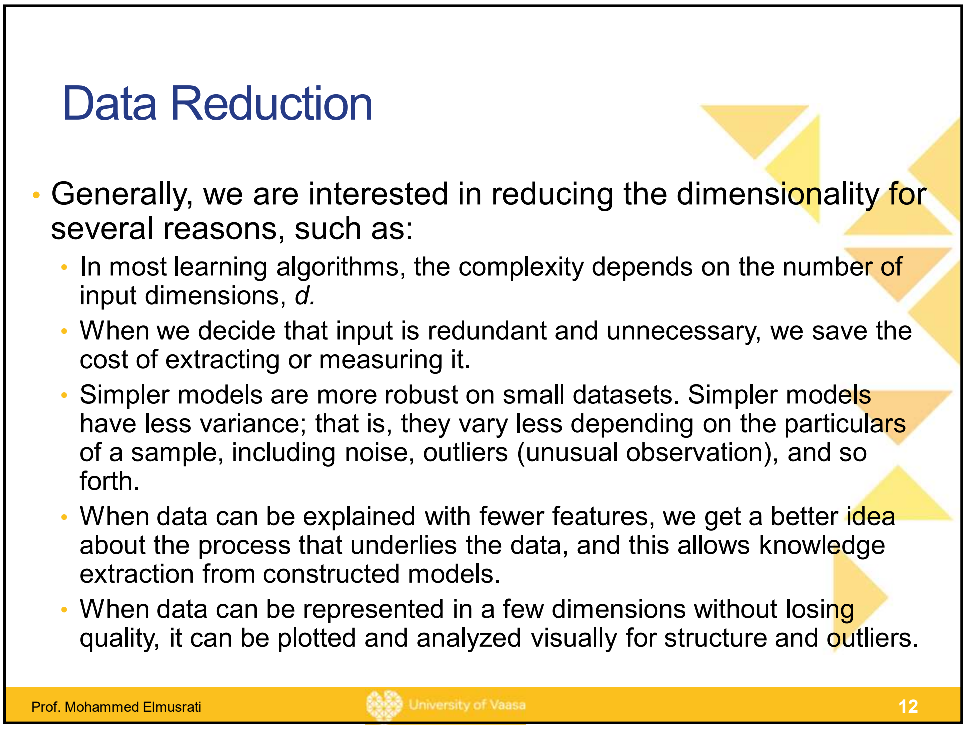


Data Reduction

- In all machine learning algorithms, it is rather essential to use only informative data and minimize redundancy or linearly related data.
- For example, assume we have 10000 inputs of data sets, and each data set consists of 100 attributes. Hence, we have 1 million data units, and our algorithms should work out of them. Usually, data are noisy, and data processing (e.g., regression, classification, etc..) will require high computation power as well as sophisticated algorithms.
- However, if 70 attributes are just linearly related to the other 30, hence those will not add any value to the results. Nevertheless, they can complicate the learning process.
- Therefore, it is crucial to remove any useless data (junk) inputs and reduce the dimension of the problem.

Prof. Mohammed Elmusrati University of Vaasa 11

11

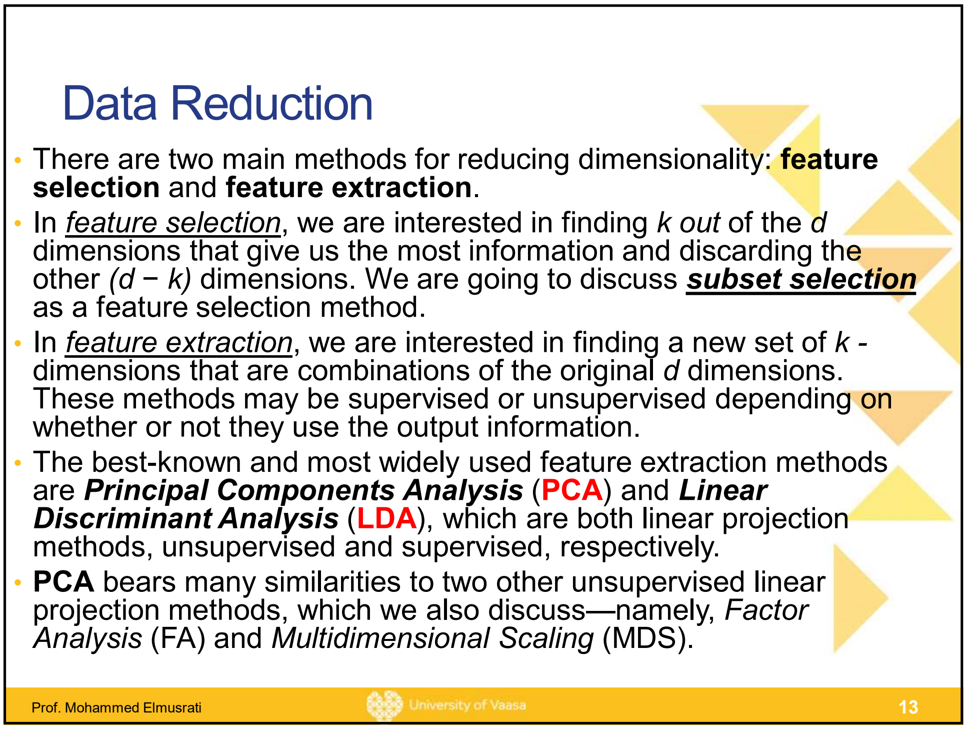


Data Reduction

- Generally, we are interested in reducing the dimensionality for several reasons, such as:
  - In most learning algorithms, the complexity depends on the number of input dimensions, *d.*
  - When we decide that input is redundant and unnecessary, we save the cost of extracting or measuring it.
  - Simpler models are more robust on small datasets. Simpler models have less variance; that is, they vary less depending on the particulars of a sample, including noise, outliers (unusual observation), and so forth.
  - When data can be explained with fewer features, we get a better idea about the process that underlies the data, and this allows knowledge extraction from constructed models.
  - When data can be represented in a few dimensions without losing quality, it can be plotted and analyzed visually for structure and outliers.

Prof. Mohammed Elmusrati University of Vaasa 12

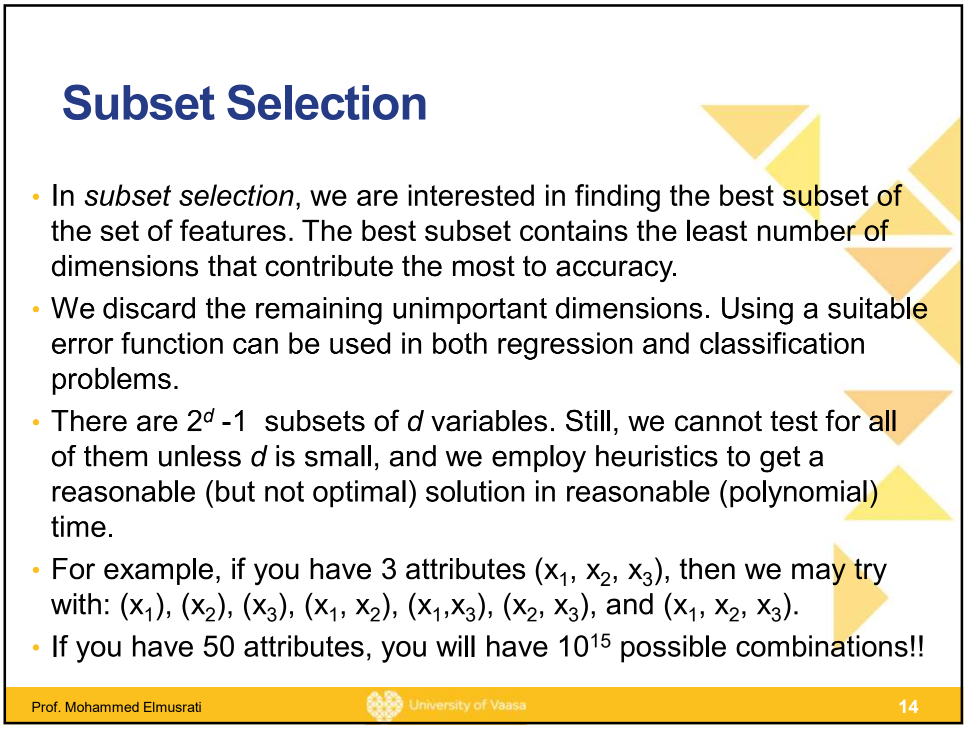
12



Data Reduction

- There are two main methods for reducing dimensionality: **feature selection** and **feature extraction**.
- In *feature selection*, we are interested in finding *k out* of the *d* dimensions that give us the most information and discarding the other *$(d - k)$* dimensions. We are going to discuss ***subset selection*** as a feature selection method.
- In *feature extraction*, we are interested in finding a new set of *k* -dimensions that are combinations of the original *d* dimensions. These methods may be supervised or unsupervised depending on whether or not they use the output information.
- The best-known and most widely used feature extraction methods are ***Principal Components Analysis*** (**PCA**) and ***Linear Discriminant Analysis*** (**LDA**), which are both linear projection methods, unsupervised and supervised, respectively.
- **PCA** bears many similarities to two other unsupervised linear projection methods, which we also discuss—namely, *Factor Analysis* (FA) and *Multidimensional Scaling* (MDS).

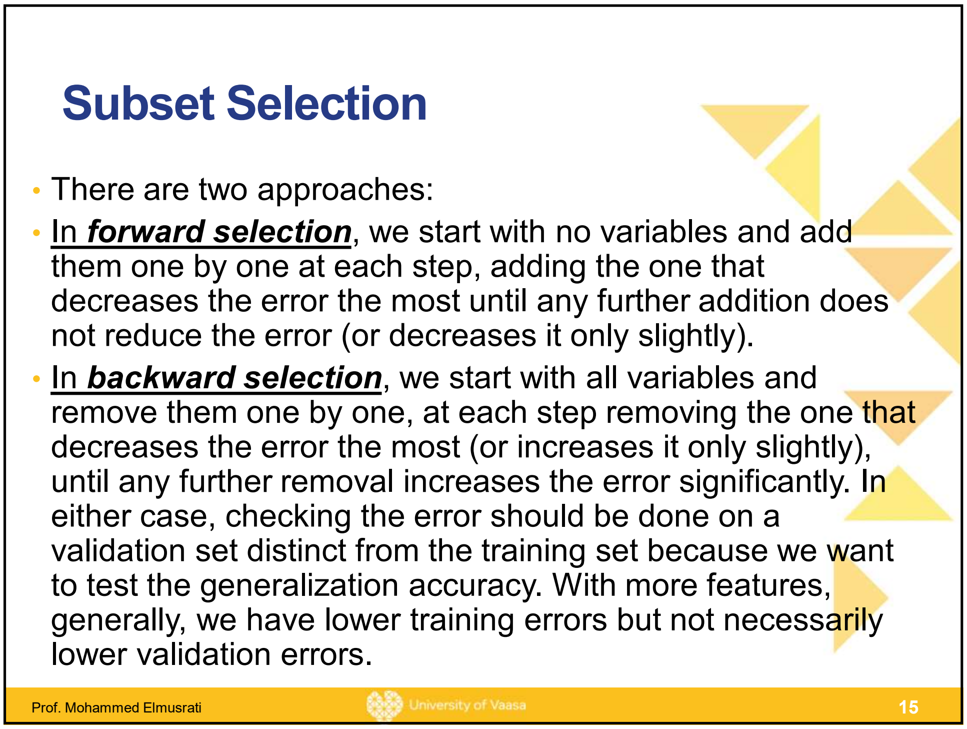
13



Subset Selection

- In *subset selection*, we are interested in finding the best subset of the set of features. The best subset contains the least number of dimensions that contribute the most to accuracy.
- We discard the remaining unimportant dimensions. Using a suitable error function can be used in both regression and classification problems.
- There are 2^d -1 subsets of *d* variables. Still, we cannot test for all of them unless *d* is small, and we employ heuristics to get a reasonable (but not optimal) solution in reasonable (polynomial) time.
- For example, if you have 3 attributes (x_1, x_2, x_3), then we may try with: (x_1), (x_2), (x_3), (x_1, x_2), (x_1,x_3), (x_2, x_3), and (x_1, x_2, x_3).
- If you have 50 attributes, you will have 10^{15} possible combinations!!

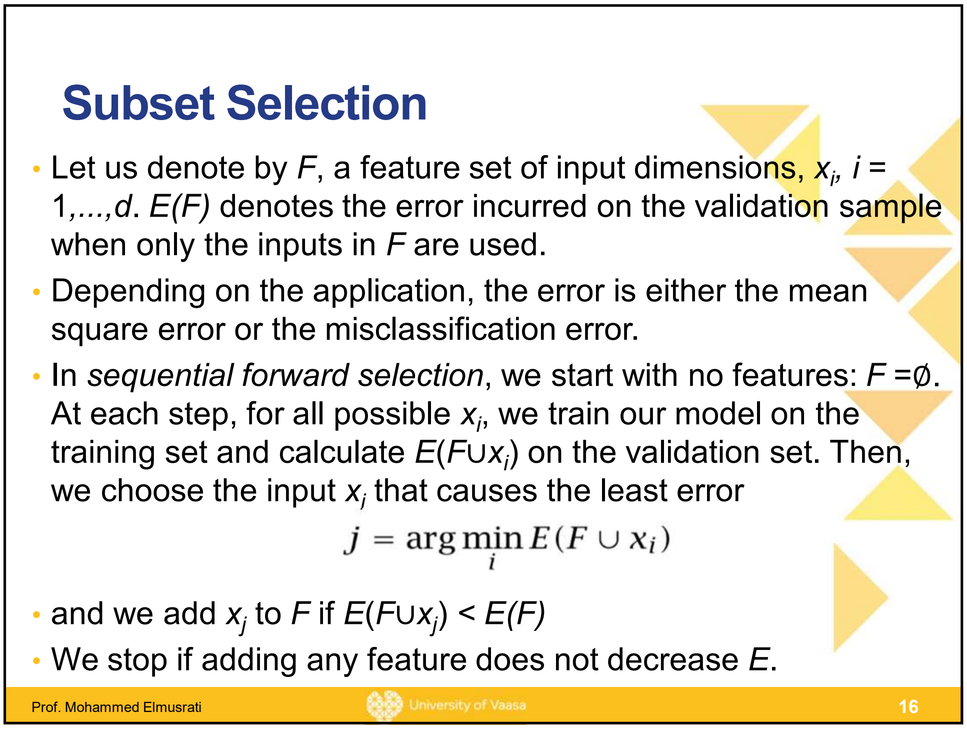
14



Subset Selection

- There are two approaches:
- In ***forward selection***, we start with no variables and add them one by one at each step, adding the one that decreases the error the most until any further addition does not reduce the error (or decreases it only slightly).
- In ***backward selection***, we start with all variables and remove them one by one, at each step removing the one that decreases the error the most (or increases it only slightly), until any further removal increases the error significantly. In either case, checking the error should be done on a validation set distinct from the training set because we want to test the generalization accuracy. With more features, generally, we have lower training errors but not necessarily lower validation errors.

15



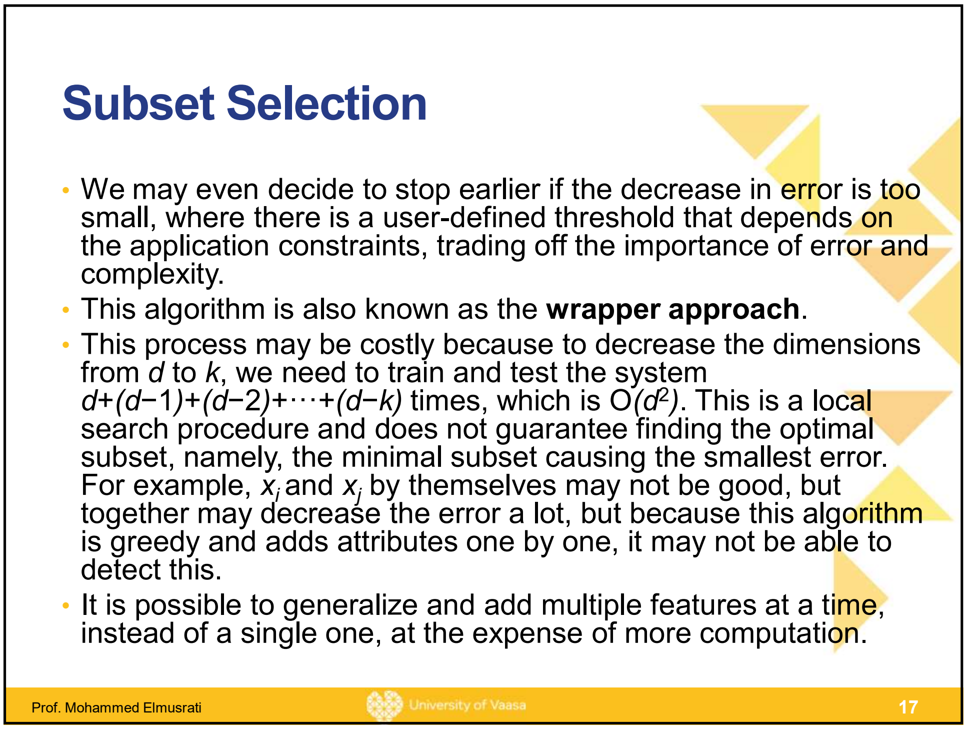
Subset Selection

- Let us denote by *F*, a feature set of input dimensions, x_i, i = 1,...,*d*. *E(F)* denotes the error incurred on the validation sample when only the inputs in *F* are used.
- Depending on the application, the error is either the mean square error or the misclassification error.
- In *sequential forward selection*, we start with no features: $F = \emptyset$. At each step, for all possible x_i, we train our model on the training set and calculate $E(F \cup x_i)$ on the validation set. Then, we choose the input x_j that causes the least error

$$j = \arg\min_i E(F \cup x_i)$$

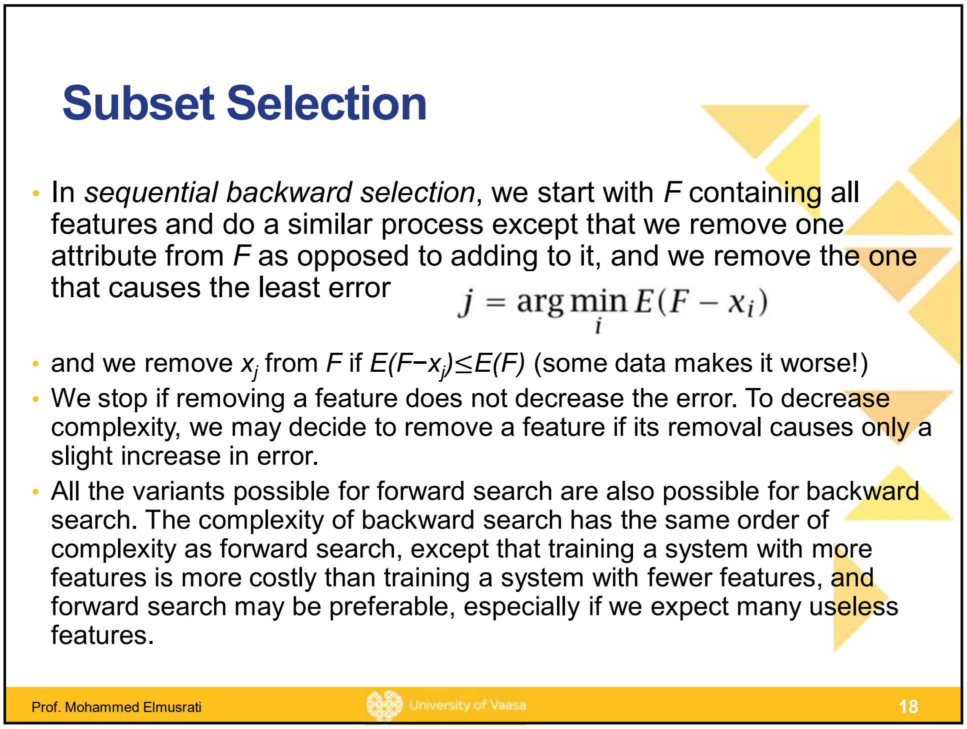
- and we add x_j to *F* if $E(F \cup x_j) < E(F)$
- We stop if adding any feature does not decrease *E*.

16



Subset Selection

- We may even decide to stop earlier if the decrease in error is too small, where there is a user-defined threshold that depends on the application constraints, trading off the importance of error and complexity.
- This algorithm is also known as the **wrapper approach**.
- This process may be costly because to decrease the dimensions from *d* to *k*, we need to train and test the system $d+(d-1)+(d-2)+\cdots+(d-k)$ times, which is $O(d^2)$. This is a local search procedure and does not guarantee finding the optimal subset, namely, the minimal subset causing the smallest error. For example, x_i and x_j by themselves may not be good, but together may decrease the error a lot, but because this algorithm is greedy and adds attributes one by one, it may not be able to detect this.
- It is possible to generalize and add multiple features at a time, instead of a single one, at the expense of more computation.

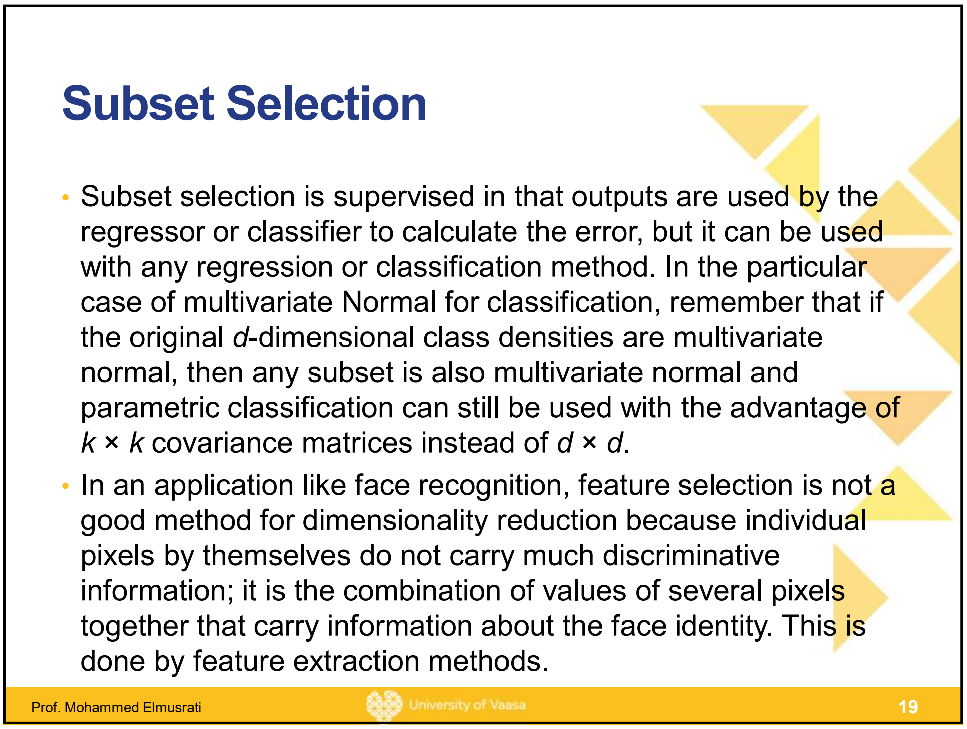


Subset Selection

- In *sequential backward selection*, we start with *F* containing all features and do a similar process except that we remove one attribute from *F* as opposed to adding to it, and we remove the one that causes the least error

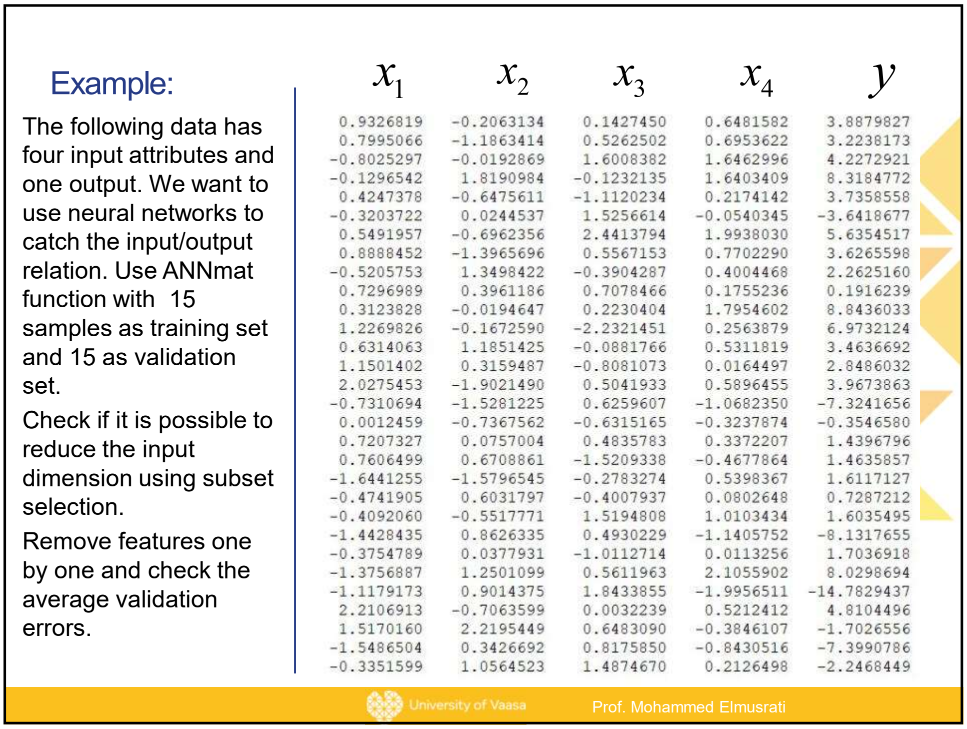
$$j = \arg\min_i E(F - x_i)$$

- and we remove x_j from *F* if $E(F-x_j) \leq E(F)$ (some data makes it worse!)
- We stop if removing a feature does not decrease the error. To decrease complexity, we may decide to remove a feature if its removal causes only a slight increase in error.
- All the variants possible for forward search are also possible for backward search. The complexity of backward search has the same order of complexity as forward search, except that training a system with more features is more costly than training a system with fewer features, and forward search may be preferable, especially if we expect many useless features.



Subset Selection

- Subset selection is supervised in that outputs are used by the regressor or classifier to calculate the error, but it can be used with any regression or classification method. In the particular case of multivariate Normal for classification, remember that if the original *d*-dimensional class densities are multivariate normal, then any subset is also multivariate normal and parametric classification can still be used with the advantage of *k* × *k* covariance matrices instead of *d* × *d*.
- In an application like face recognition, feature selection is not a good method for dimensionality reduction because individual pixels by themselves do not carry much discriminative information; it is the combination of values of several pixels together that carry information about the face identity. This is done by feature extraction methods.



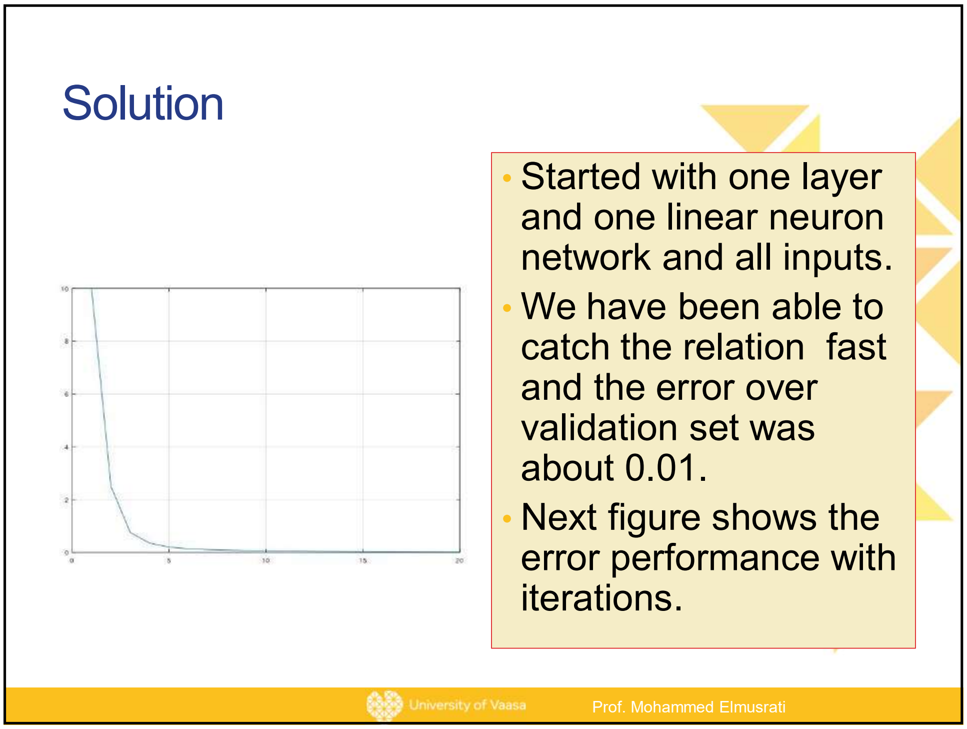
Example:

The following data has four input attributes and one output. We want to use neural networks to catch the input/output relation. Use ANNmat function with 15 samples as training set and 15 as validation set.

Check if it is possible to reduce the input dimension using subset selection.

Remove features one by one and check the average validation errors.

x_1	x_2	x_3	x_4	y
0.9326819	-0.2063134	0.1427450	0.6481582	3.8879827
0.7995066	-1.1863414	0.5262502	0.6953622	3.2238173
-0.8025297	-0.0192869	1.6008382	1.6462996	4.2272921
-0.1296542	1.8190984	-0.1232135	1.6403409	8.3184772
0.4247378	-0.6475611	-1.1120234	0.2174142	3.7358558
-0.3203722	0.0244537	1.5256614	-0.0540345	-3.6418677
0.5491957	-0.6962356	2.4413794	1.9938030	5.6354517
0.8888452	-1.3965696	0.5567153	0.7702290	3.6265598
-0.5205753	1.3498422	-0.3904287	0.4004468	2.2625160
0.7296989	0.3961186	0.7078466	0.1755236	0.1916239
0.3123828	-0.0194647	0.2230404	1.7954602	8.8436033
1.2269826	-0.1672590	-2.2321451	0.2563879	6.9732124
0.6314063	1.1851425	-0.0881766	0.5311819	3.4636692
1.1501402	0.3159487	-0.8081073	0.0164497	2.8486032
2.0275453	-1.9021490	0.5041933	0.5896455	3.9673863
-0.7310694	-1.5281225	0.6259607	-1.0682350	-7.3241656
0.0012459	-0.7367562	-0.6315165	-0.3237874	-0.3546580
0.7207327	0.0757004	0.4835783	0.3372207	1.4396796
0.7606499	0.6708861	-1.5209338	-0.4677864	1.4635857
-1.6441255	-1.5796545	-0.2783274	0.5398367	1.6117127
-0.4741905	0.6031797	-0.4007937	0.0802648	0.7287212
-0.4092060	-0.5517771	1.5194808	1.0103434	1.6035495
-1.4428435	0.8626335	0.4930229	-1.1405752	-8.1317655
-0.3754789	0.0377931	-1.0112714	0.0113256	1.7036918
-1.3756887	1.2501099	0.5611963	2.1055902	8.0298694
-1.1179173	0.9014375	1.8433855	-1.9956511	-14.7829437
2.2106913	-0.7063599	0.0032239	0.5212412	4.8104496
1.5170160	2.2195449	0.6483090	-0.3846107	-1.7026556
-1.5486504	0.3426692	0.8175850	-0.8430516	-7.3990786
-0.3351599	1.0564523	1.4874670	0.2126498	-2.2468449

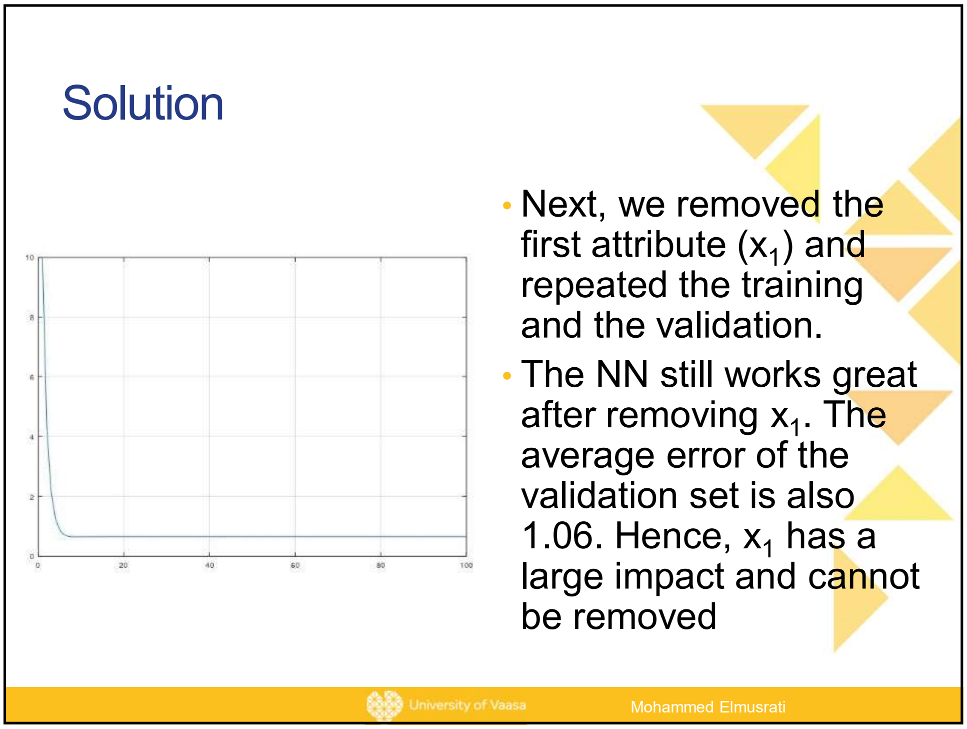


Solution

- Started with one layer and one linear neuron network and all inputs.
- We have been able to catch the relation fast and the error over validation set was about 0.01.
- Next figure shows the error performance with iterations.

University of Vaasa Prof. Mohammed Elmusrati

21



Solution

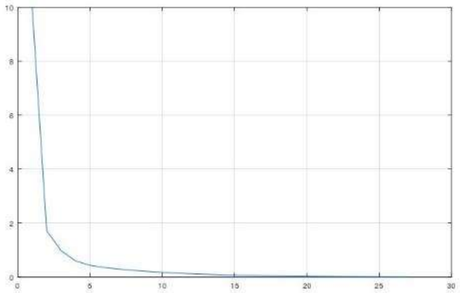
- Next, we removed the first attribute (x_1) and repeated the training and the validation.
- The NN still works great after removing x_1. The average error of the validation set is also 1.06. Hence, x_1 has a large impact and cannot be removed

University of Vaasa Mohammed Elmusrati

22

Solution

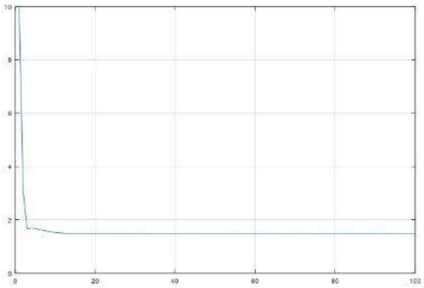
- Now, we returned x_1 and removed x_2 from the learning and validation set.
- The average error over the validation set was again very small, around 0.01.
- This means that x_2 has no considerable effect



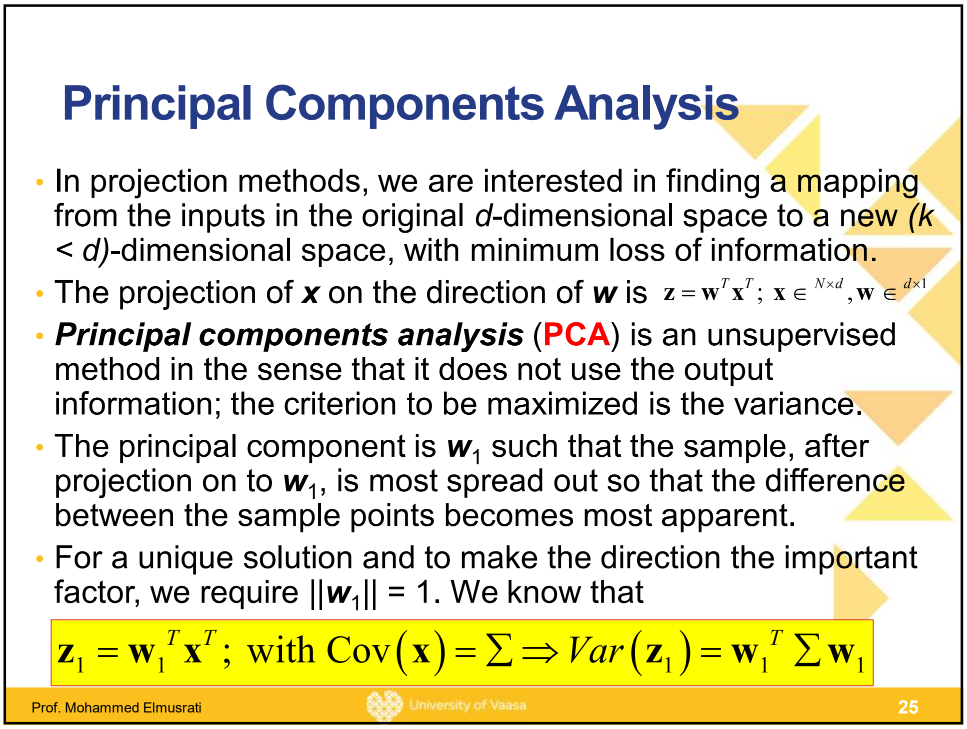
23

Solution

- Here, we remove the third input, and check the training as well as the validation.
- The error is large after removing the third input. This means that the third input is critical and should be kept.
- We continuo in this way!



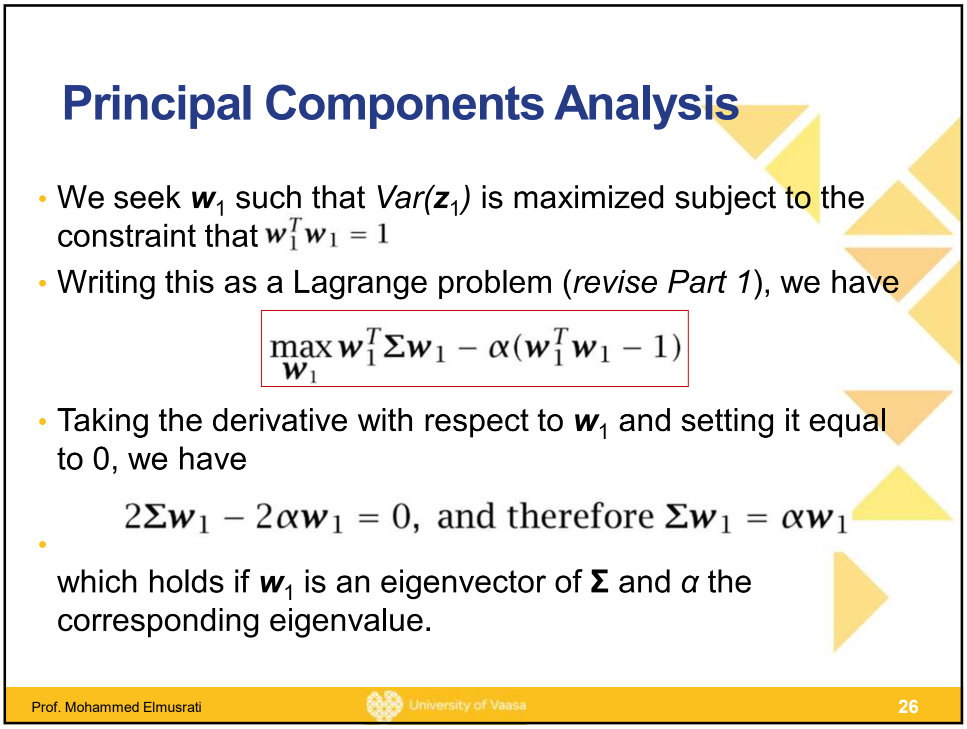
24



Principal Components Analysis

- In projection methods, we are interested in finding a mapping from the inputs in the original *d*-dimensional space to a new *(k < d)*-dimensional space, with minimum loss of information.
- The projection of ***x*** on the direction of ***w*** is $\mathbf{z}=\mathbf{w}^T\mathbf{x}^T;\ \mathbf{x}\in{}^{N\times d},\mathbf{w}\in{}^{d\times 1}$
- ***Principal components analysis*** (**PCA**) is an unsupervised method in the sense that it does not use the output information; the criterion to be maximized is the variance.
- The principal component is $\boldsymbol{w}_1$ such that the sample, after projection on to $\boldsymbol{w}_1$, is most spread out so that the difference between the sample points becomes most apparent.
- For a unique solution and to make the direction the important factor, we require $||\boldsymbol{w}_1|| = 1$. We know that

$$\mathbf{z}_1=\mathbf{w}_1^T\mathbf{x}^T;\ \text{with Cov}(\mathbf{x})=\Sigma\Rightarrow Var(\mathbf{z}_1)=\mathbf{w}_1^T\Sigma\mathbf{w}_1$$



Principal Components Analysis

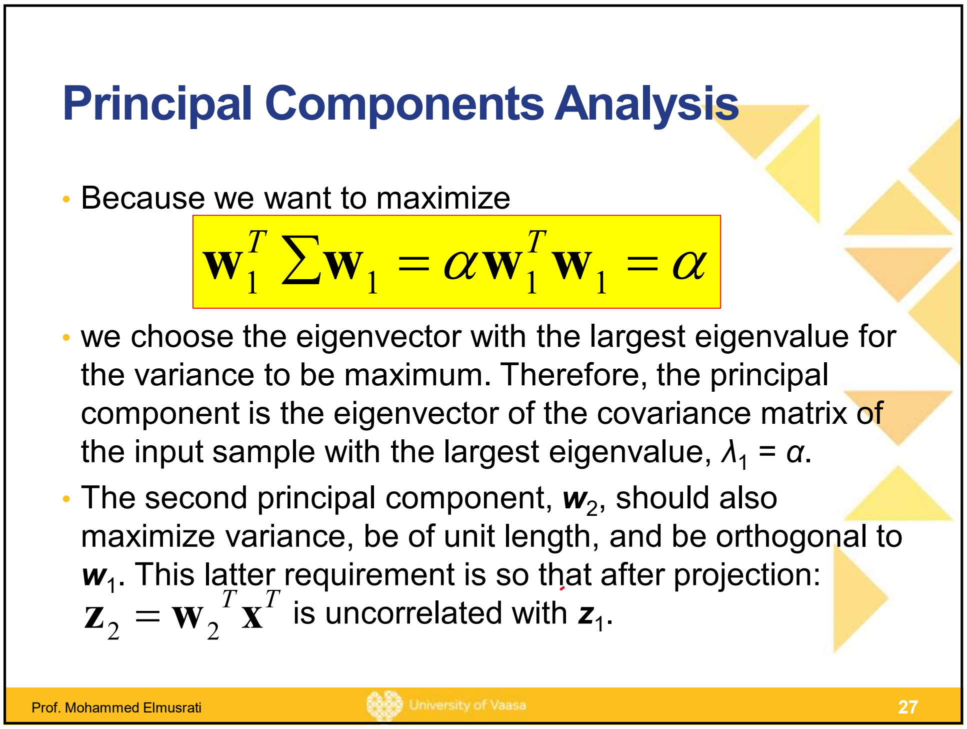
- We seek $\boldsymbol{w}_1$ such that *Var(**z**$_1$)* is maximized subject to the constraint that $\boldsymbol{w}_1^T\boldsymbol{w}_1=1$
- Writing this as a Lagrange problem (*revise Part 1*), we have

$$\max_{\boldsymbol{w}_1}\boldsymbol{w}_1^T\Sigma\boldsymbol{w}_1-\alpha(\boldsymbol{w}_1^T\boldsymbol{w}_1-1)$$

- Taking the derivative with respect to $\boldsymbol{w}_1$ and setting it equal to 0, we have
-

$$2\Sigma\boldsymbol{w}_1-2\alpha\boldsymbol{w}_1=0,\ \text{and therefore}\ \Sigma\boldsymbol{w}_1=\alpha\boldsymbol{w}_1$$

which holds if $\boldsymbol{w}_1$ is an eigenvector of **Σ** and *α* the corresponding eigenvalue.



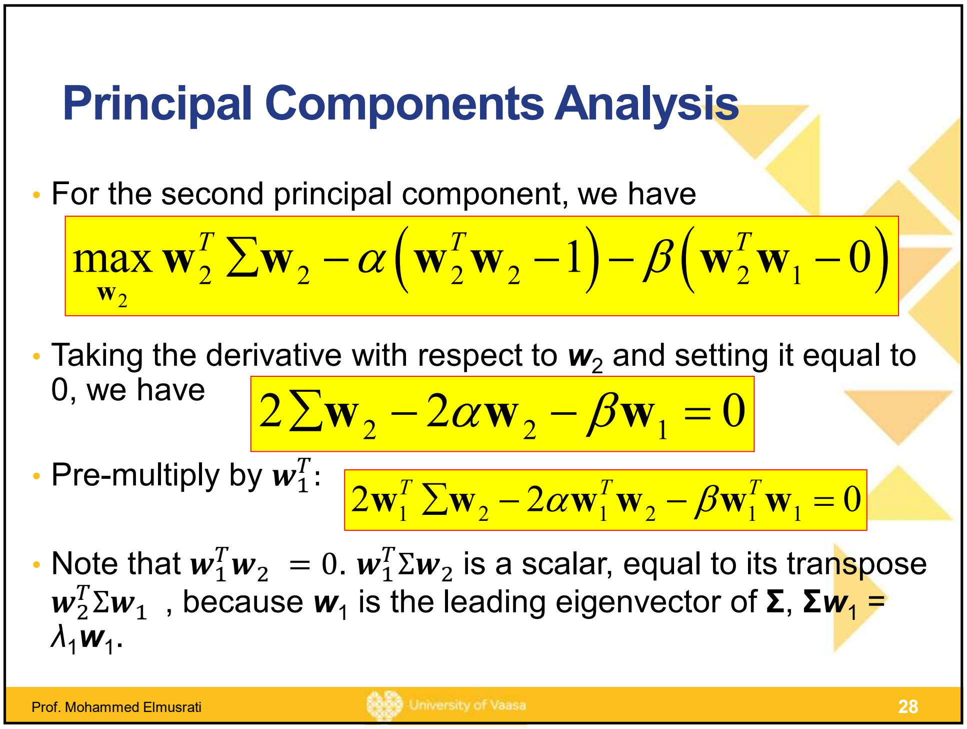
Principal Components Analysis

- Because we want to maximize

$$\mathbf{w}_1^T \Sigma \mathbf{w}_1 = \alpha \mathbf{w}_1^T \mathbf{w}_1 = \alpha$$

- we choose the eigenvector with the largest eigenvalue for the variance to be maximum. Therefore, the principal component is the eigenvector of the covariance matrix of the input sample with the largest eigenvalue, $\lambda_1 = \alpha$.
- The second principal component, $\boldsymbol{w}_2$, should also maximize variance, be of unit length, and be orthogonal to $\boldsymbol{w}_1$. This latter requirement is so that after projection: $\mathbf{z}_2 = \mathbf{w}_2{}^T \mathbf{x}^T$ is uncorrelated with $\boldsymbol{z}_1$.

27



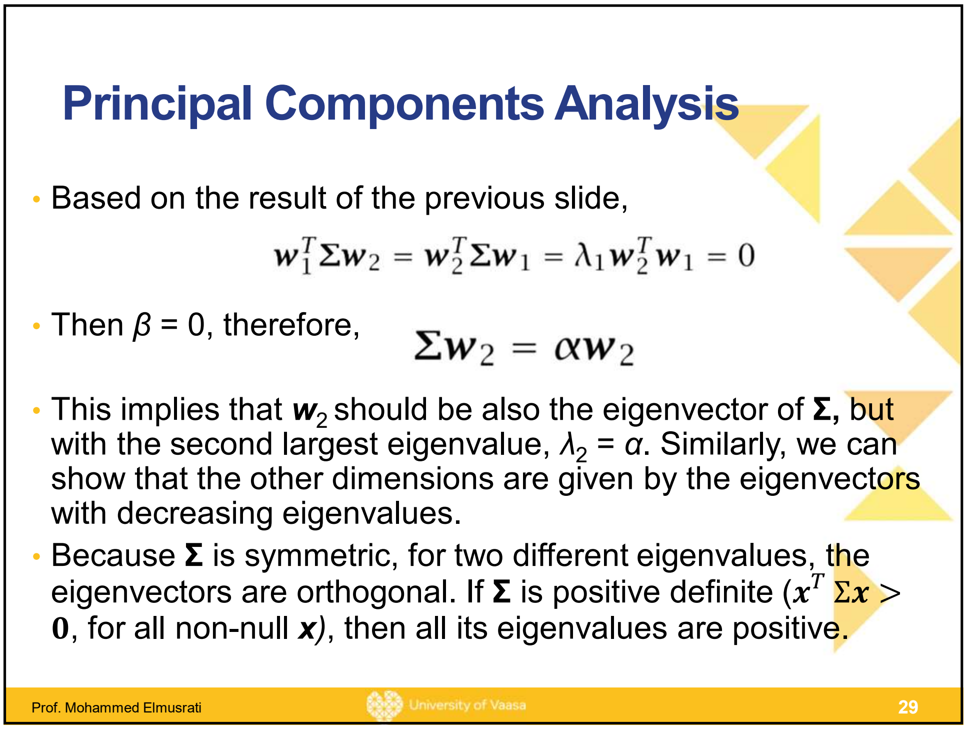
Principal Components Analysis

- For the second principal component, we have

$$\max_{\mathbf{w}_2} \mathbf{w}_2^T \Sigma \mathbf{w}_2 - \alpha\left(\mathbf{w}_2^T \mathbf{w}_2 - 1\right) - \beta\left(\mathbf{w}_2^T \mathbf{w}_1 - 0\right)$$

- Taking the derivative with respect to $\boldsymbol{w}_2$ and setting it equal to 0, we have $2\Sigma \mathbf{w}_2 - 2\alpha \mathbf{w}_2 - \beta \mathbf{w}_1 = 0$
- Pre-multiply by $\boldsymbol{w}_1^T$: $2\mathbf{w}_1^T \Sigma \mathbf{w}_2 - 2\alpha \mathbf{w}_1^T \mathbf{w}_2 - \beta \mathbf{w}_1^T \mathbf{w}_1 = 0$
- Note that $\boldsymbol{w}_1^T \boldsymbol{w}_2 = 0$. $\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_2$ is a scalar, equal to its transpose $\boldsymbol{w}_2^T \Sigma \boldsymbol{w}_1$, because $\boldsymbol{w}_1$ is the leading eigenvector of **Σ**, $\boldsymbol{\Sigma w}_1 = \lambda_1 \boldsymbol{w}_1$.

28



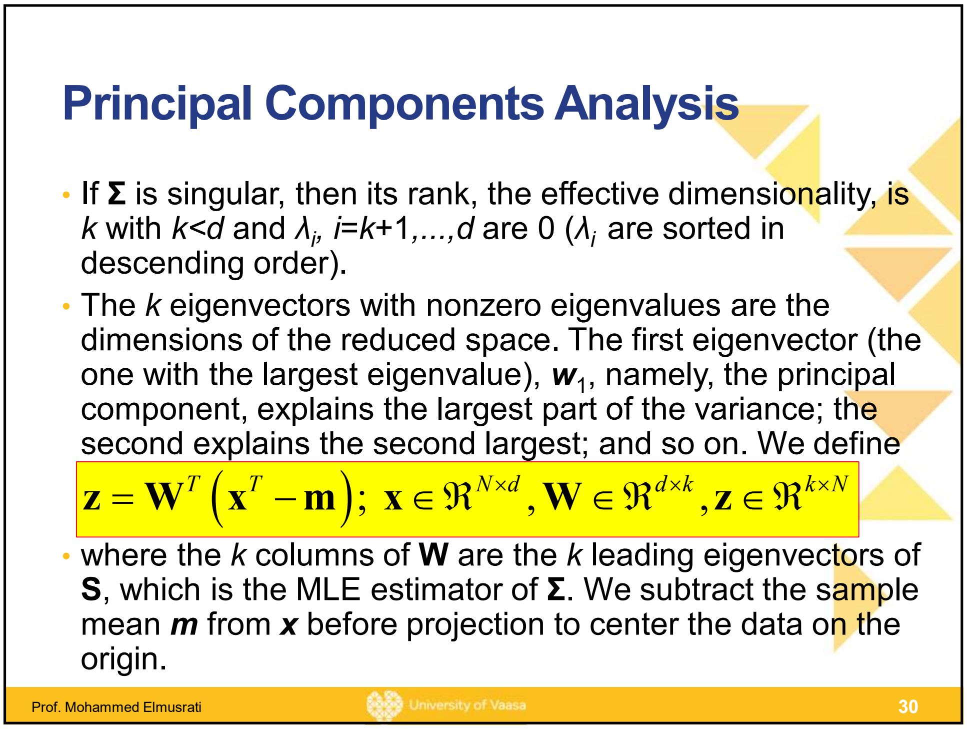
Principal Components Analysis

- Based on the result of the previous slide,

$$\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_2 = \boldsymbol{w}_2^T \Sigma \boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_2^T \boldsymbol{w}_1 = 0$$

- Then $\beta = 0$, therefore, $\Sigma \boldsymbol{w}_2 = \alpha \boldsymbol{w}_2$
- This implies that $\boldsymbol{w}_2$ should be also the eigenvector of **Σ, but** with the second largest eigenvalue, $\lambda_2 = \alpha$. Similarly, we can show that the other dimensions are given by the eigenvectors with decreasing eigenvalues.
- Because **Σ** is symmetric, for two different eigenvalues, the eigenvectors are orthogonal. If **Σ** is positive definite ($\boldsymbol{x}^T\ \Sigma\boldsymbol{x} > \mathbf{0}$, for all non-null $\boldsymbol{x}$), then all its eigenvalues are positive.

Prof. Mohammed Elmusrati University of Vaasa 29



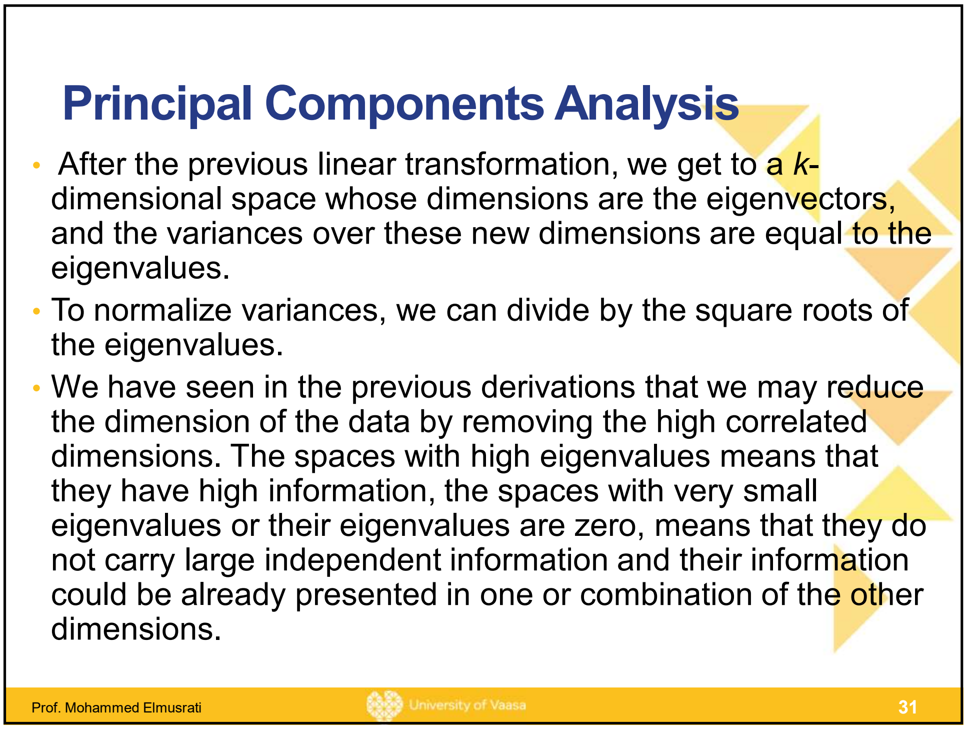
Principal Components Analysis

- If **Σ** is singular, then its rank, the effective dimensionality, is *k* with *k*<*d* and λ_i, *i*=*k*+1,...,*d* are 0 (λ_i are sorted in descending order).
- The *k* eigenvectors with nonzero eigenvalues are the dimensions of the reduced space. The first eigenvector (the one with the largest eigenvalue), $\boldsymbol{w}_1$, namely, the principal component, explains the largest part of the variance; the second explains the second largest; and so on. We define

$$\mathbf{z} = \mathbf{W}^T\left(\mathbf{x}^T - \mathbf{m}\right);\ \mathbf{x} \in \Re^{N\times d}, \mathbf{W} \in \Re^{d\times k}, \mathbf{z} \in \Re^{k\times N}$$

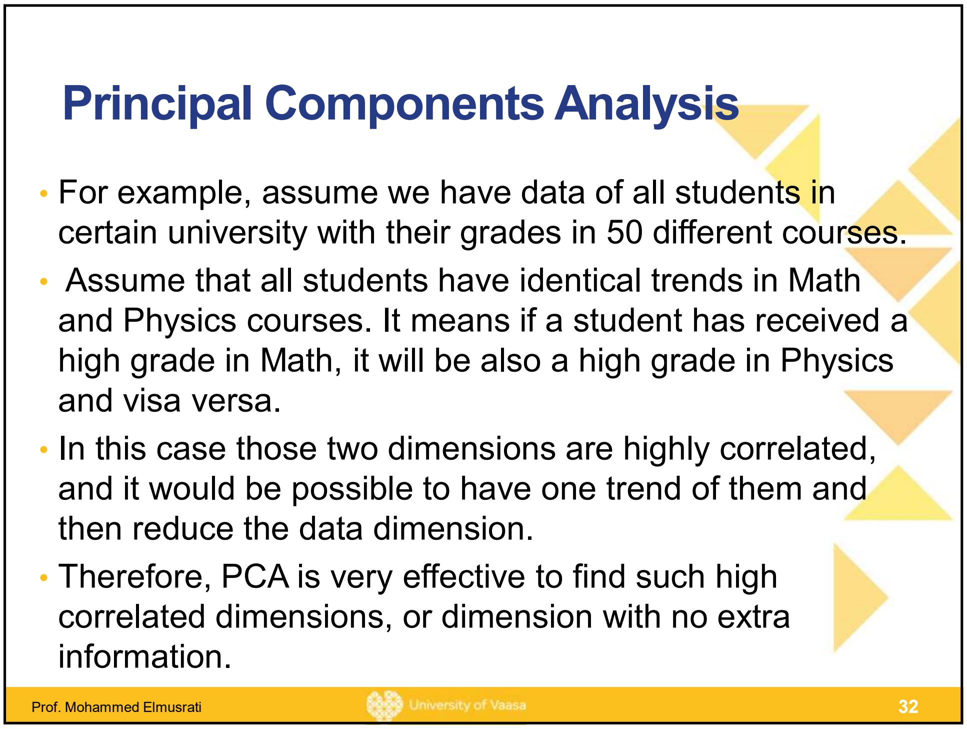
- where the *k* columns of **W** are the *k* leading eigenvectors of **S**, which is the MLE estimator of **Σ**. We subtract the sample mean ***m*** from ***x*** before projection to center the data on the origin.

Prof. Mohammed Elmusrati University of Vaasa 30



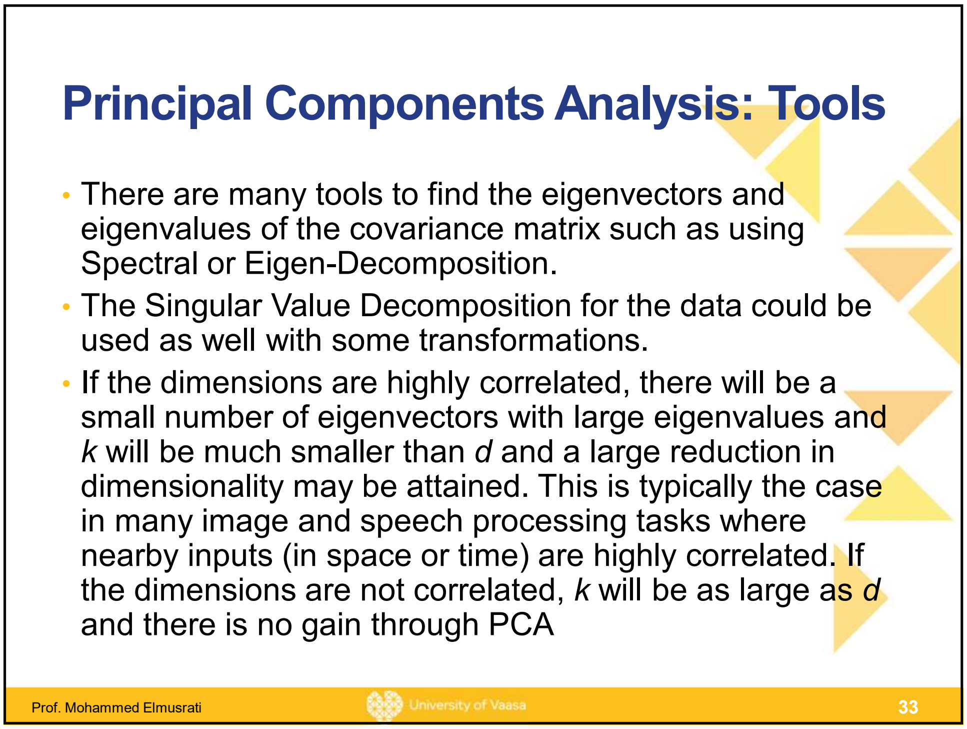
Principal Components Analysis

- After the previous linear transformation, we get to a *k*-dimensional space whose dimensions are the eigenvectors, and the variances over these new dimensions are equal to the eigenvalues.
- To normalize variances, we can divide by the square roots of the eigenvalues.
- We have seen in the previous derivations that we may reduce the dimension of the data by removing the high correlated dimensions. The spaces with high eigenvalues means that they have high information, the spaces with very small eigenvalues or their eigenvalues are zero, means that they do not carry large independent information and their information could be already presented in one or combination of the other dimensions.



Principal Components Analysis

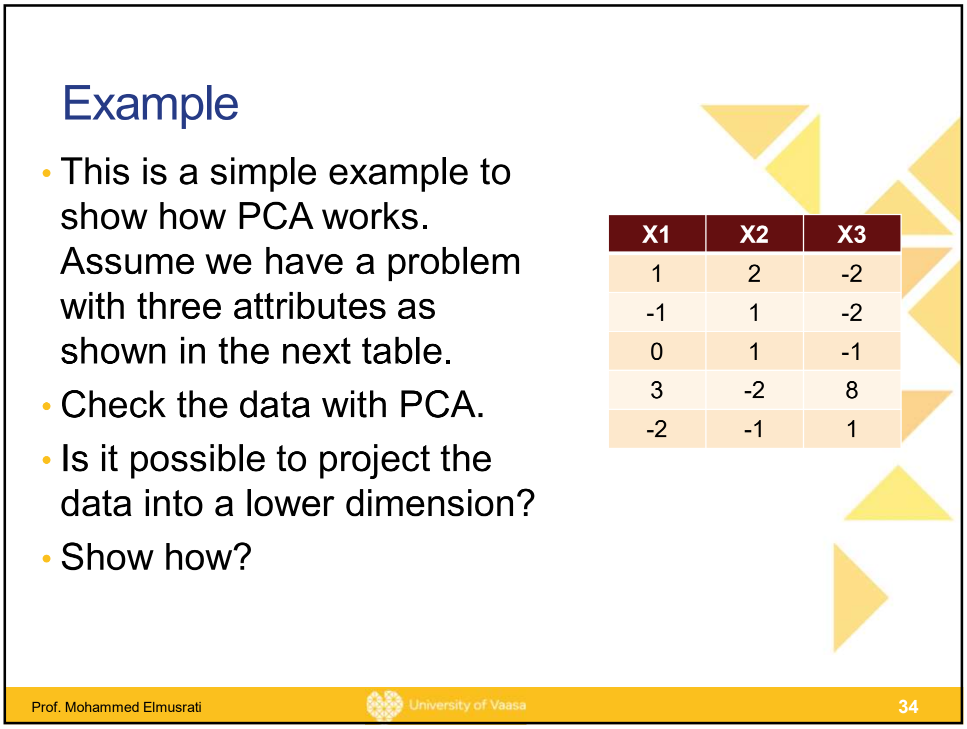
- For example, assume we have data of all students in certain university with their grades in 50 different courses.
- Assume that all students have identical trends in Math and Physics courses. It means if a student has received a high grade in Math, it will be also a high grade in Physics and visa versa.
- In this case those two dimensions are highly correlated, and it would be possible to have one trend of them and then reduce the data dimension.
- Therefore, PCA is very effective to find such high correlated dimensions, or dimension with no extra information.



Principal Components Analysis: Tools

- There are many tools to find the eigenvectors and eigenvalues of the covariance matrix such as using Spectral or Eigen-Decomposition.
- The Singular Value Decomposition for the data could be used as well with some transformations.
- If the dimensions are highly correlated, there will be a small number of eigenvectors with large eigenvalues and *k* will be much smaller than *d* and a large reduction in dimensionality may be attained. This is typically the case in many image and speech processing tasks where nearby inputs (in space or time) are highly correlated. If the dimensions are not correlated, *k* will be as large as *d* and there is no gain through PCA

33



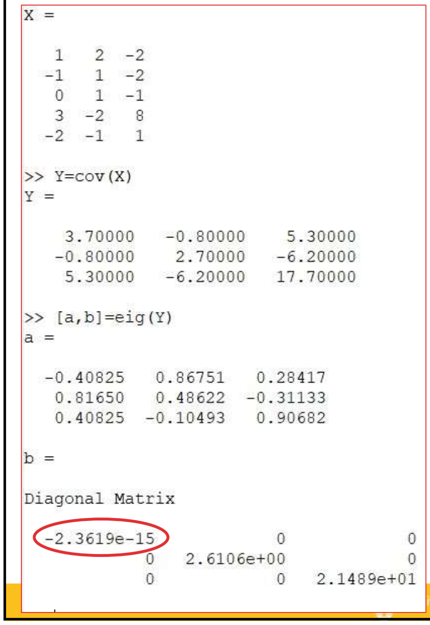
Example

- This is a simple example to show how PCA works. Assume we have a problem with three attributes as shown in the next table.
- Check the data with PCA.
- Is it possible to project the data into a lower dimension?
- Show how?

X1	X2	X3
1	2	-2
-1	1	-2
0	1	-1
3	-2	8
-2	-1	1

34

Example



```
X =

   1   2  -2
  -1   1  -2
   0   1  -1
   3  -2   8
  -2  -1   1

>> Y=cov(X)
Y =

    3.70000   -0.80000    5.30000
   -0.80000    2.70000   -6.20000
    5.30000   -6.20000   17.70000

>> [a,b]=eig(Y)
a =

  -0.40825   0.86751   0.28417
   0.81650   0.48622  -0.31133
   0.40825  -0.10493   0.90682

b =

Diagonal Matrix

  -2.3619e-15              0              0
            0     2.6106e+00              0
            0              0     2.1489e+01
```

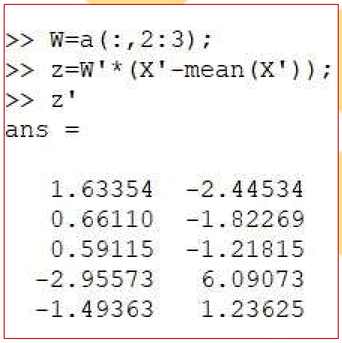
- We need to compute the covariance matrix of the input data.
- This example is quite small in size and can be solved with hand calculation.
- However, we used Octave to find the eigenvalues and eigenvectors as shown.

Example

- From the results, it is obvious that we can reduce the dimension at least to size 2. The first eigenvalue is almost zero, this means that there is high correlation in the data.
- We can project the data into two dimensions without losing any information.
- As shown in the analysis the projection should be done over the eigenvectors of the largest eigenvalues, i.e., in this example the second and the third eigenvalues.
- Nest slide shows how to do that.

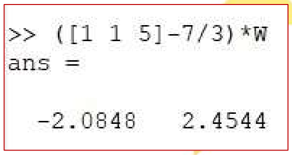
Example

- We have multiplied the original data by a matrix with the second and the third eigenvectors, as shown in the figure.
- If we have a new data set, like [1 1 5], we can project it into our new data set, as also shown in the figure.
- These two dimensions still carry all information in the original 3-dimensions data.
- If it is OK to lose some information, we can still project the data to only one dimension by considering only the third eigenvector. It has a much higher eigenvalue than the second one.



```
>> W=a(:,2:3);
>> z=W'*(X'-mean(X'));
>> z'
ans =

   1.63354  -2.44534
   0.66110  -1.82269
   0.59115  -1.21815
  -2.95573   6.09073
  -1.49363   1.23625
```

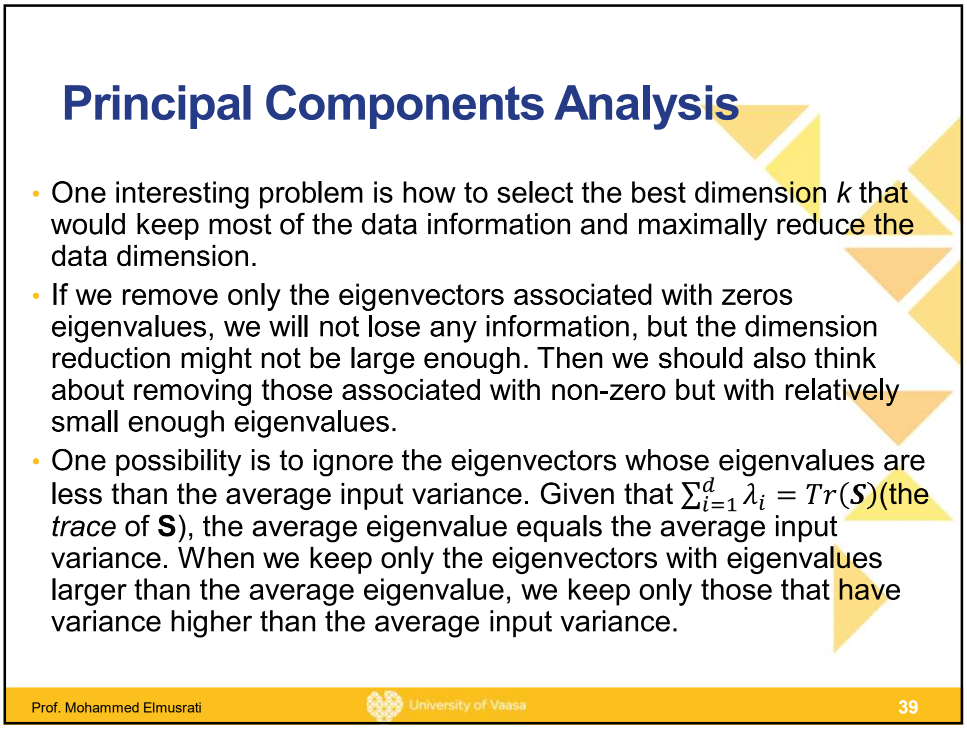


```
>> ([1 1 5]-7/3)*W
ans =

  -2.0848    2.4544
```

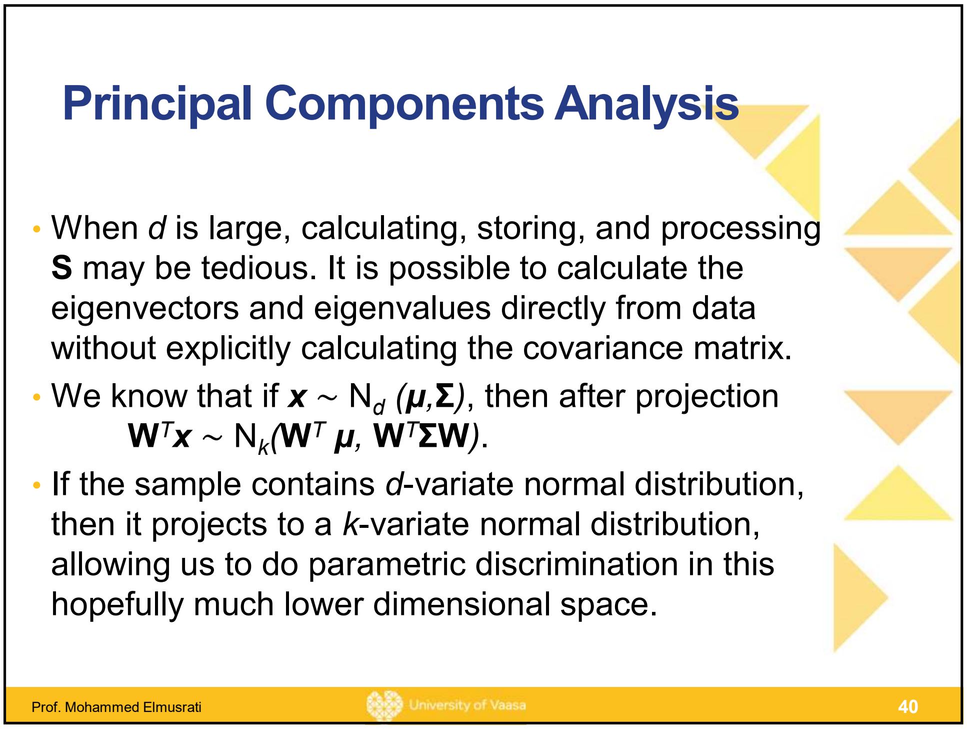
Example:

- Data_Part5_1 is a data file of 200 samples in 12 dimensions. Use PCA to find the minimum dimension size we could use without losing information? If we could allow some information to be lost in order to minimize the dimension. What is your proposed dimension. Regenerate the data with feature extraction in the new dimension size.
- Solution: First, we should estimate the covariance matrix S which will be 12x12 dimension. Then we diagonalizable the matrix to find their eigenvalues and eigenvectors.
- We found the eigenvalues vector in descending order: {**6.10**, **4.38**, **3.35**, **1.64**, **0.35**, **0.33**, **0.00, 0.00, 0.00,0.00,0.00,0.00**}
- Hence it is clear that the losses dimension is 6. We can rebuild the data for complete feature extraction with only 6 dimensions instead of 12. If it is possible to have a slight loss, then we can also ignore two more dimensions with the smallest nonzero eigenvalues. Hence, we have only 4 dimensions.



Principal Components Analysis

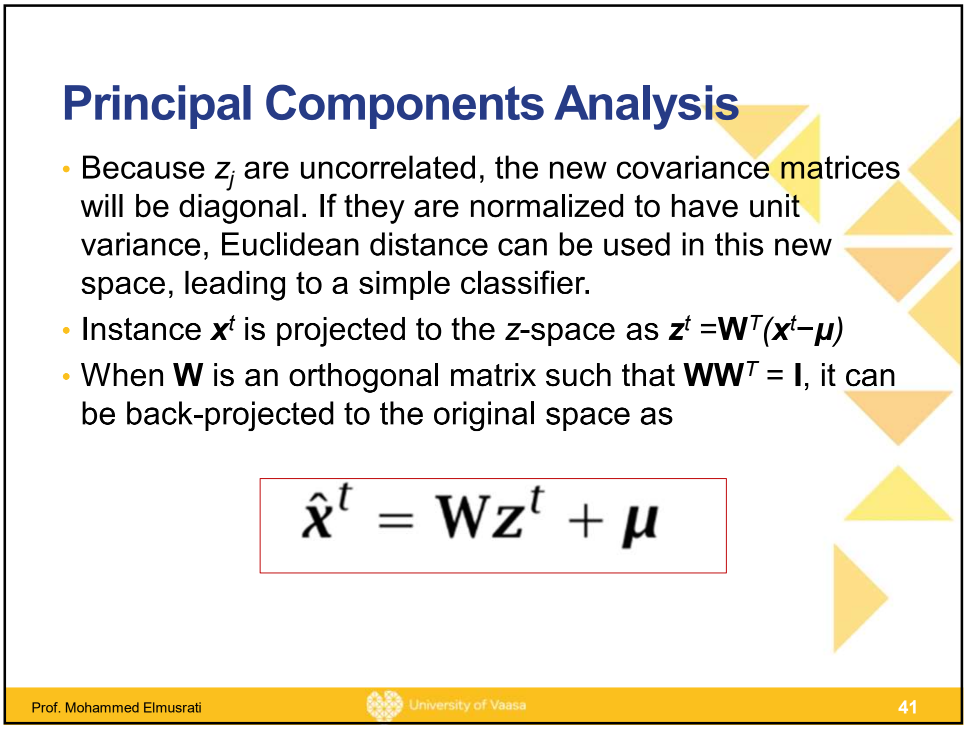
- One interesting problem is how to select the best dimension *k* that would keep most of the data information and maximally reduce the data dimension.
- If we remove only the eigenvectors associated with zeros eigenvalues, we will not lose any information, but the dimension reduction might not be large enough. Then we should also think about removing those associated with non-zero but with relatively small enough eigenvalues.
- One possibility is to ignore the eigenvectors whose eigenvalues are less than the average input variance. Given that $\sum_{i=1}^{d}\lambda_i = Tr(\boldsymbol{S})$(the *trace* of **S**), the average eigenvalue equals the average input variance. When we keep only the eigenvectors with eigenvalues larger than the average eigenvalue, we keep only those that have variance higher than the average input variance.



Principal Components Analysis

- When *d* is large, calculating, storing, and processing **S** may be tedious. It is possible to calculate the eigenvectors and eigenvalues directly from data without explicitly calculating the covariance matrix.
- We know that if $\boldsymbol{x} \sim N_d(\boldsymbol{\mu},\boldsymbol{\Sigma})$, then after projection

  $\mathbf{W}^T\boldsymbol{x} \sim N_k(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\boldsymbol{\Sigma}\mathbf{W})$.
- If the sample contains *d*-variate normal distribution, then it projects to a *k*-variate normal distribution, allowing us to do parametric discrimination in this hopefully much lower dimensional space.

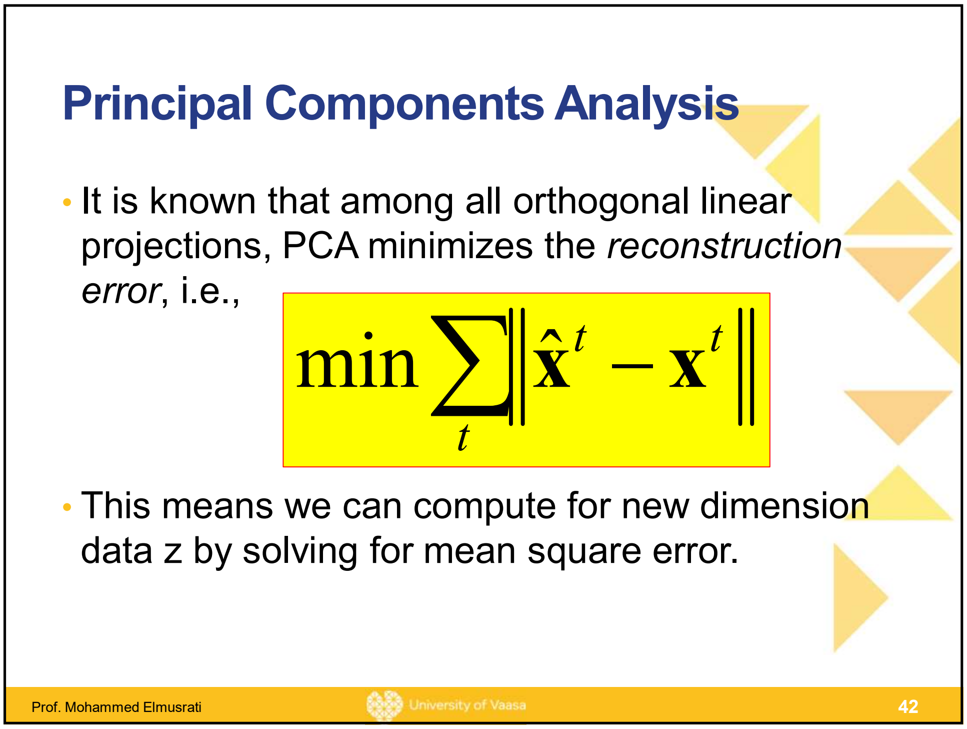


Principal Components Analysis

- Because z_j are uncorrelated, the new covariance matrices will be diagonal. If they are normalized to have unit variance, Euclidean distance can be used in this new space, leading to a simple classifier.
- Instance $\boldsymbol{x}^t$ is projected to the z-space as $\boldsymbol{z}^t = \mathbf{W}^T(\boldsymbol{x}^t - \boldsymbol{\mu})$
- When $\mathbf{W}$ is an orthogonal matrix such that $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, it can be back-projected to the original space as

$$\boxed{\hat{\mathbf{x}}^t = \mathbf{W}\mathbf{z}^t + \boldsymbol{\mu}}$$

41



Principal Components Analysis

- It is known that among all orthogonal linear projections, PCA minimizes the *reconstruction error*, i.e.,

$$\boxed{\min \sum_t \left\| \hat{\mathbf{x}}^t - \mathbf{x}^t \right\|}$$

- This means we can compute for new dimension data z by solving for mean square error.

42

Feature Embedding

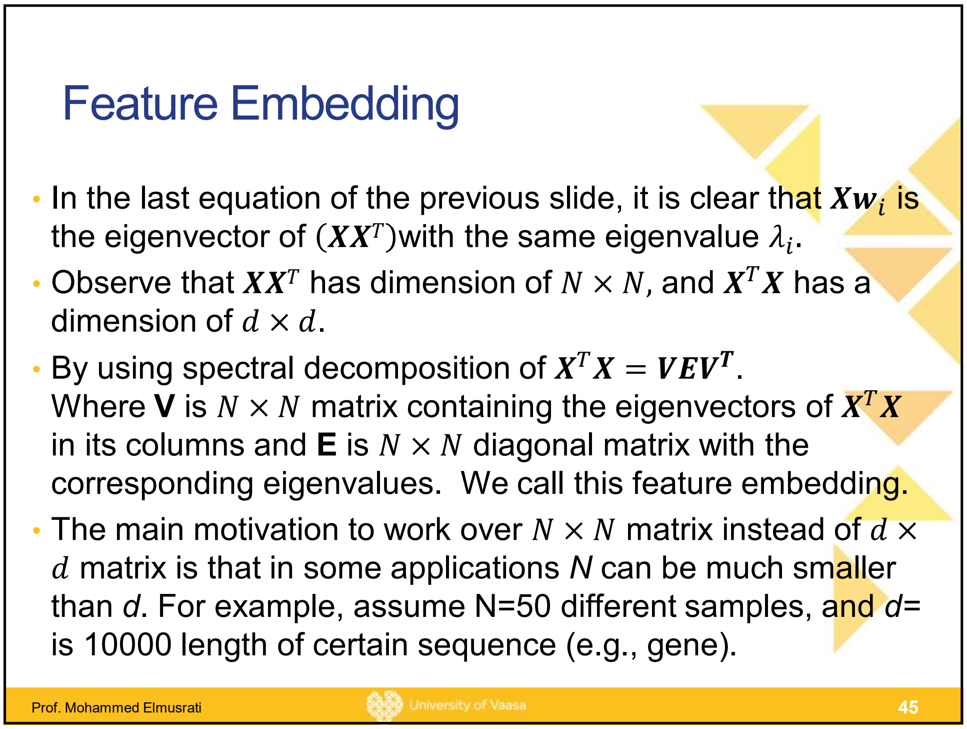
- Our data matrix is *X* with a dimension of $N \times d$, where *N* is the number of instances and *d* is the input dimensionality.
- The covariance matrix of $\boldsymbol{x}$ is $d \times d$, and in the case of ergodic process it could be estimated from the data as: *X^TX/N*, if X is centered on having zero means. Otherwise, we subtract their average values.
- PCA uses the eigenvectors of X^TX. It computes $X^TX=\boldsymbol{WDW}^T$, where $\boldsymbol{W}$ is $d \times d$ and contains the eigenvectors of X^TX in its columns and $\boldsymbol{D}$ is a $d \times d$ diagonal matrix with the correspondent eigenvalues. We assume that the eigenvalues are sorted in ***D,*** starting from the largest absolute value to the smallest.

Feature Embedding

- If we want to reduce the dimensionality from *d* to *k*, where *k<d*, then as shown in PCA, we may construct the transformation matrix **W** with the first *k* eigenvectors associated with the highest eigenvalue.
- Let's denote each eigenvector as $\boldsymbol{w}_i$ and its associated eigenvalue as λ_i, where *i=1,..,k.*
- We map to the new *k-dimensional* space by taking a dot product of the original inputs with the eigenvectors as

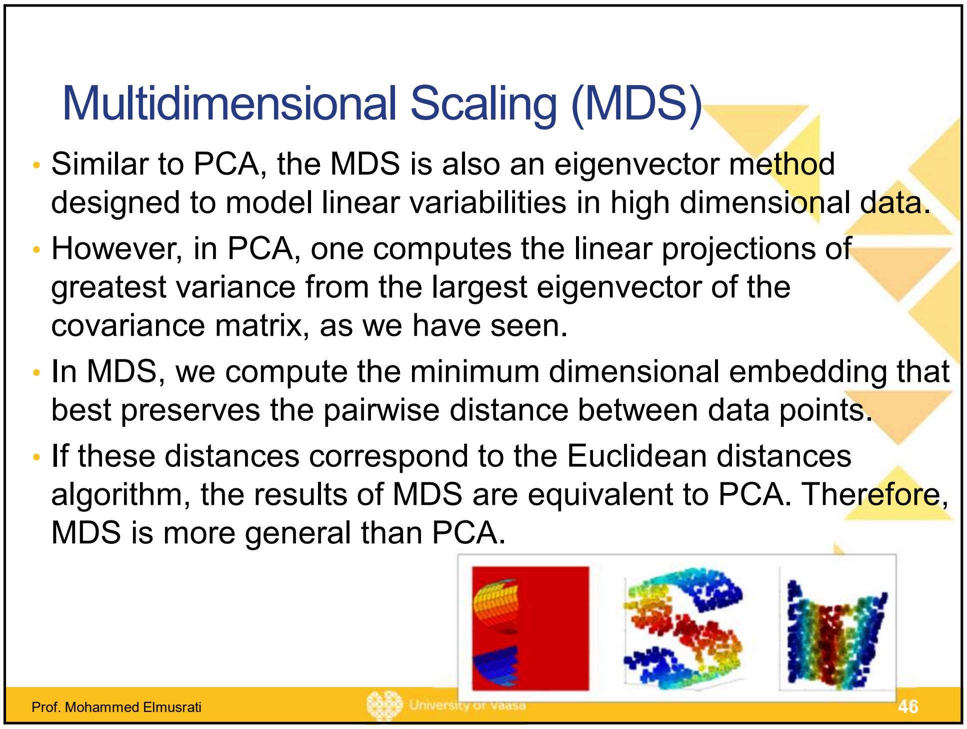
$$z_i^t = \boldsymbol{w}_i^t \boldsymbol{x}^t, i = 1, \dots, k; t = 1, \dots, N$$

- Given that λ_i and $\boldsymbol{w}_i$ are the eigenvalues and eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$, for any *i*≤*k*, we have: $(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$, pre-multiplying by $\boldsymbol{X}$, we find $(\boldsymbol{X}\boldsymbol{X}^T)\boldsymbol{X}\boldsymbol{w}_i = \lambda_i \boldsymbol{X}\boldsymbol{w}_i$



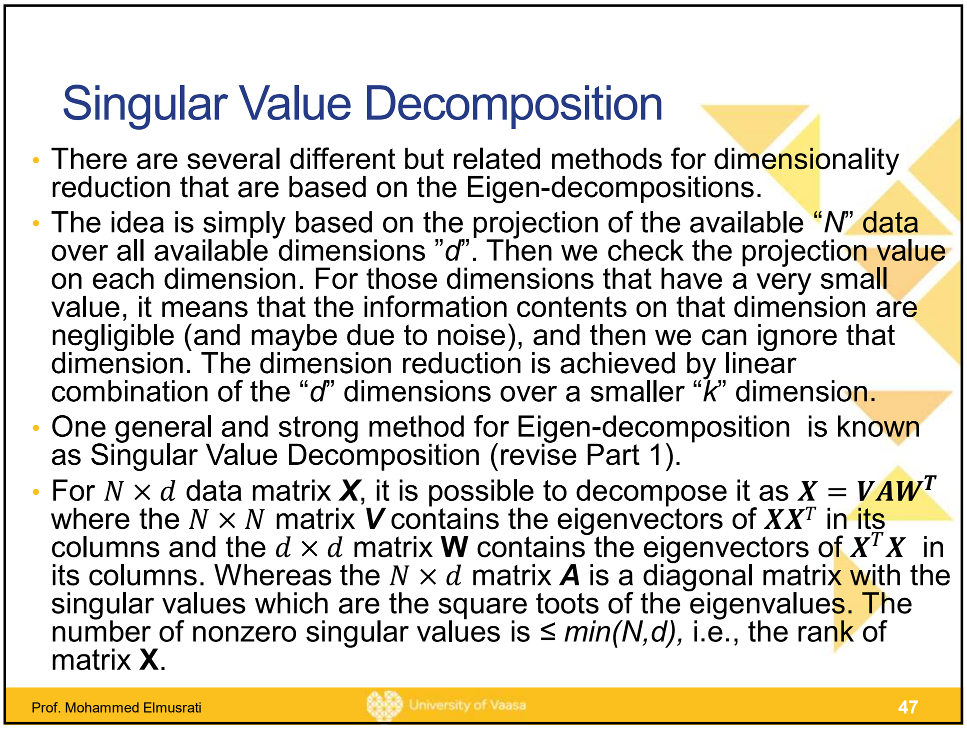
Feature Embedding

- In the last equation of the previous slide, it is clear that $\boldsymbol{Xw}_i$ is the eigenvector of $(\boldsymbol{XX}^T)$with the same eigenvalue λ_i.
- Observe that $\boldsymbol{XX}^T$ has dimension of $N \times N$, and $\boldsymbol{X}^T\boldsymbol{X}$ has a dimension of $d \times d$.
- By using spectral decomposition of $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{VEV^T}$. Where **V** is $N \times N$ matrix containing the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ in its columns and **E** is $N \times N$ diagonal matrix with the corresponding eigenvalues. We call this feature embedding.
- The main motivation to work over $N \times N$ matrix instead of $d \times d$ matrix is that in some applications *N* can be much smaller than *d*. For example, assume N=50 different samples, and *d*= is 10000 length of certain sequence (e.g., gene).



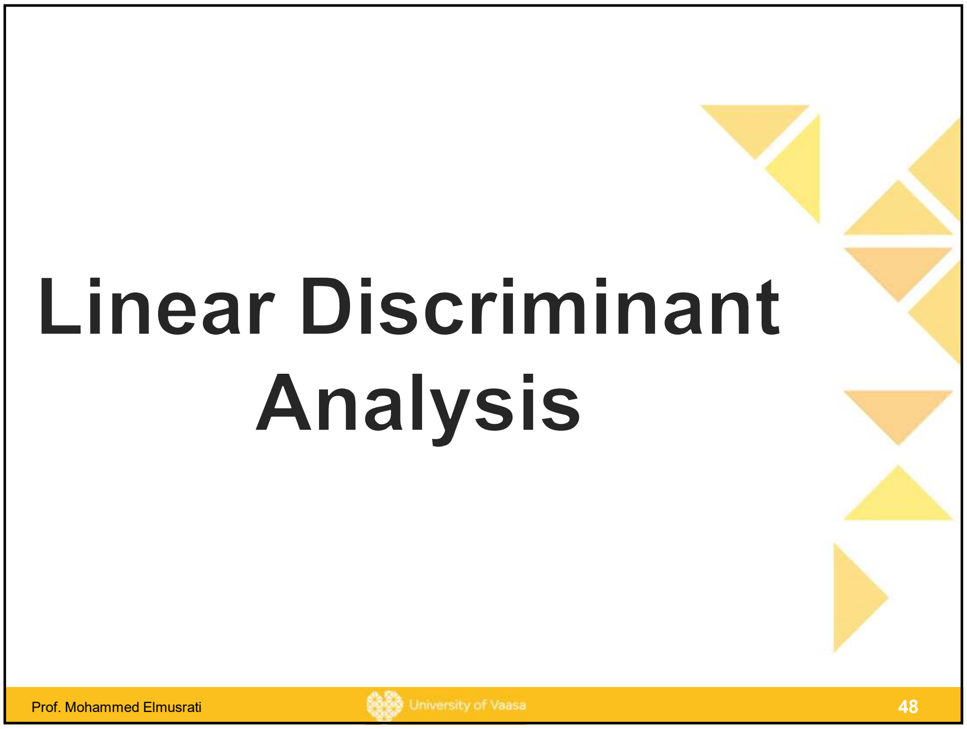
Multidimensional Scaling (MDS)

- Similar to PCA, the MDS is also an eigenvector method designed to model linear variabilities in high dimensional data.
- However, in PCA, one computes the linear projections of greatest variance from the largest eigenvector of the covariance matrix, as we have seen.
- In MDS, we compute the minimum dimensional embedding that best preserves the pairwise distance between data points.
- If these distances correspond to the Euclidean distances algorithm, the results of MDS are equivalent to PCA. Therefore, MDS is more general than PCA.

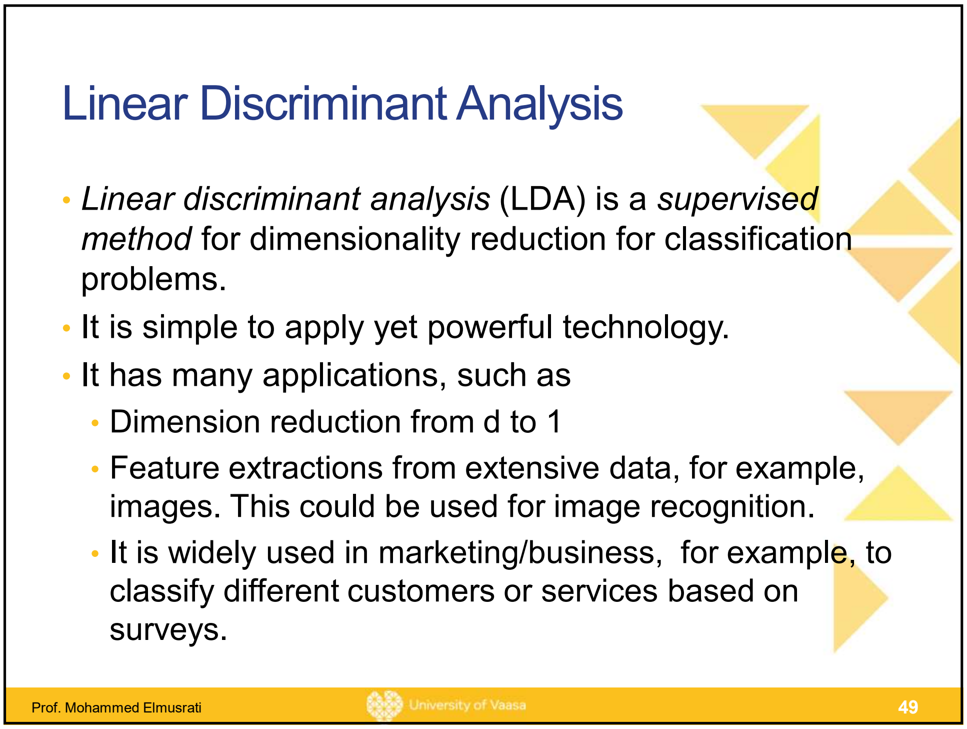


Singular Value Decomposition

- There are several different but related methods for dimensionality reduction that are based on the Eigen-decompositions.
- The idea is simply based on the projection of the available "*N*" data over all available dimensions "*d*". Then we check the projection value on each dimension. For those dimensions that have a very small value, it means that the information contents on that dimension are negligible (and maybe due to noise), and then we can ignore that dimension. The dimension reduction is achieved by linear combination of the "*d*" dimensions over a smaller "*k*" dimension.
- One general and strong method for Eigen-decomposition is known as Singular Value Decomposition (revise Part 1).
- For $N \times d$ data matrix ***X***, it is possible to decompose it as $\boldsymbol{X} = \boldsymbol{VAW^T}$ where the $N \times N$ matrix ***V*** contains the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$ in its columns and the $d \times d$ matrix **W** contains the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ in its columns. Whereas the $N \times d$ matrix ***A*** is a diagonal matrix with the singular values which are the square toots of the eigenvalues. The number of nonzero singular values is ≤ *min(N,d),* i.e., the rank of matrix **X**.

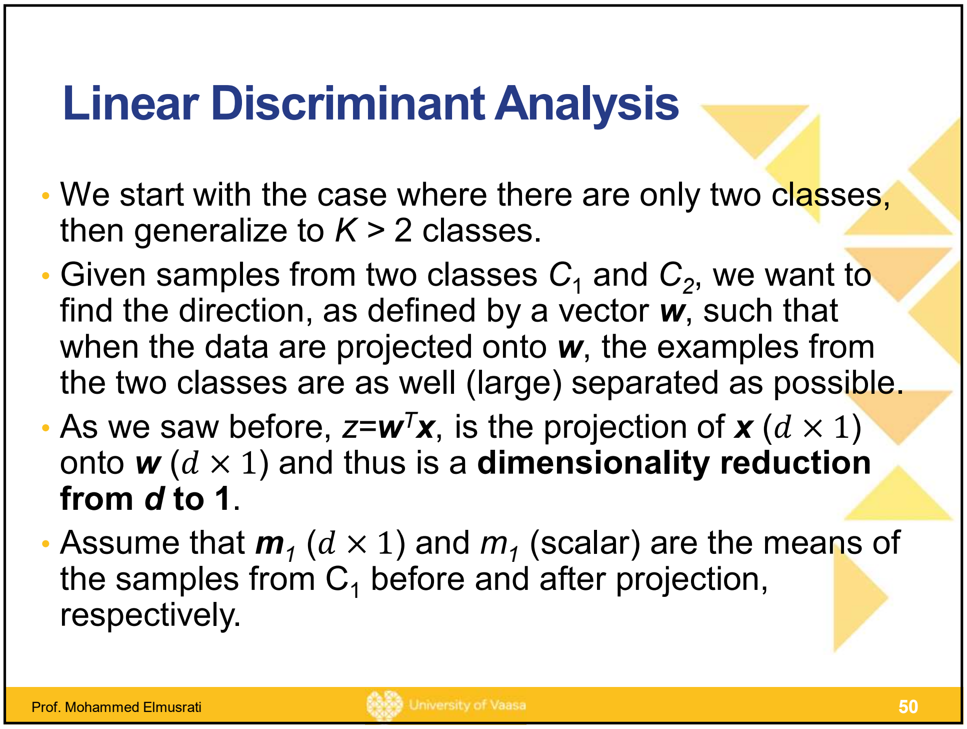


Linear Discriminant Analysis



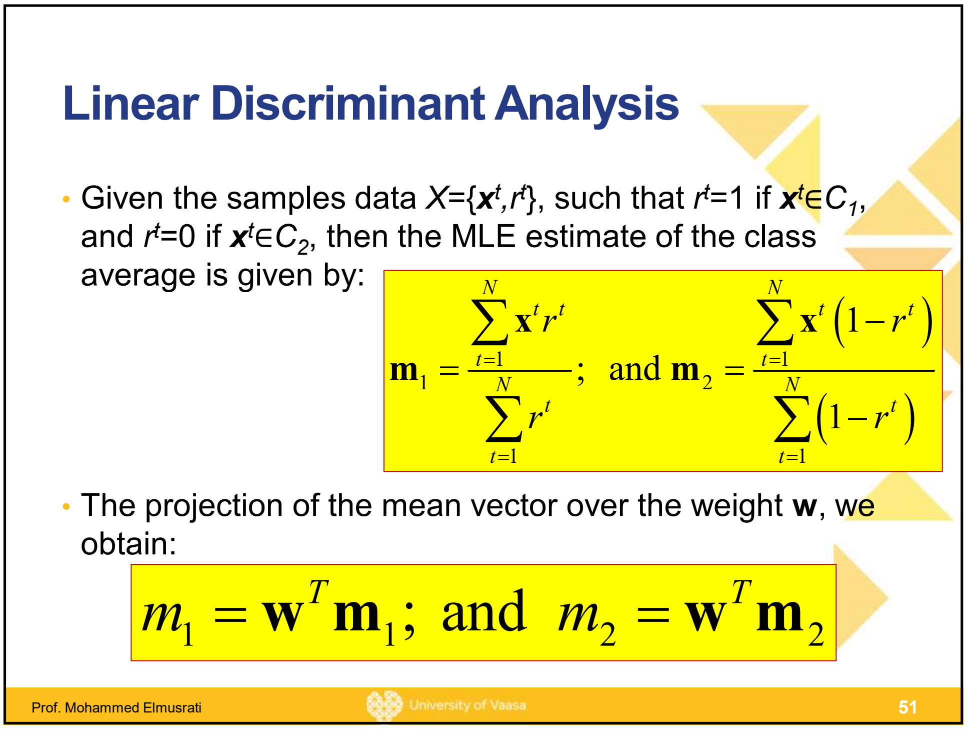
Linear Discriminant Analysis

- *Linear discriminant analysis* (LDA) is a *supervised method* for dimensionality reduction for classification problems.
- It is simple to apply yet powerful technology.
- It has many applications, such as
  - Dimension reduction from d to 1
  - Feature extractions from extensive data, for example, images. This could be used for image recognition.
  - It is widely used in marketing/business, for example, to classify different customers or services based on surveys.



Linear Discriminant Analysis

- We start with the case where there are only two classes, then generalize to $K > 2$ classes.
- Given samples from two classes C_1 and C_2, we want to find the direction, as defined by a vector $\boldsymbol{w}$, such that when the data are projected onto $\boldsymbol{w}$, the examples from the two classes are as well (large) separated as possible.
- As we saw before, $z=\boldsymbol{w}^T\boldsymbol{x}$, is the projection of $\boldsymbol{x}$ $(d \times 1)$ onto $\boldsymbol{w}$ $(d \times 1)$ and thus is a **dimensionality reduction from *d* to 1**.
- Assume that $\boldsymbol{m}_1$ $(d \times 1)$ and m_1 (scalar) are the means of the samples from C_1 before and after projection, respectively.



Linear Discriminant Analysis

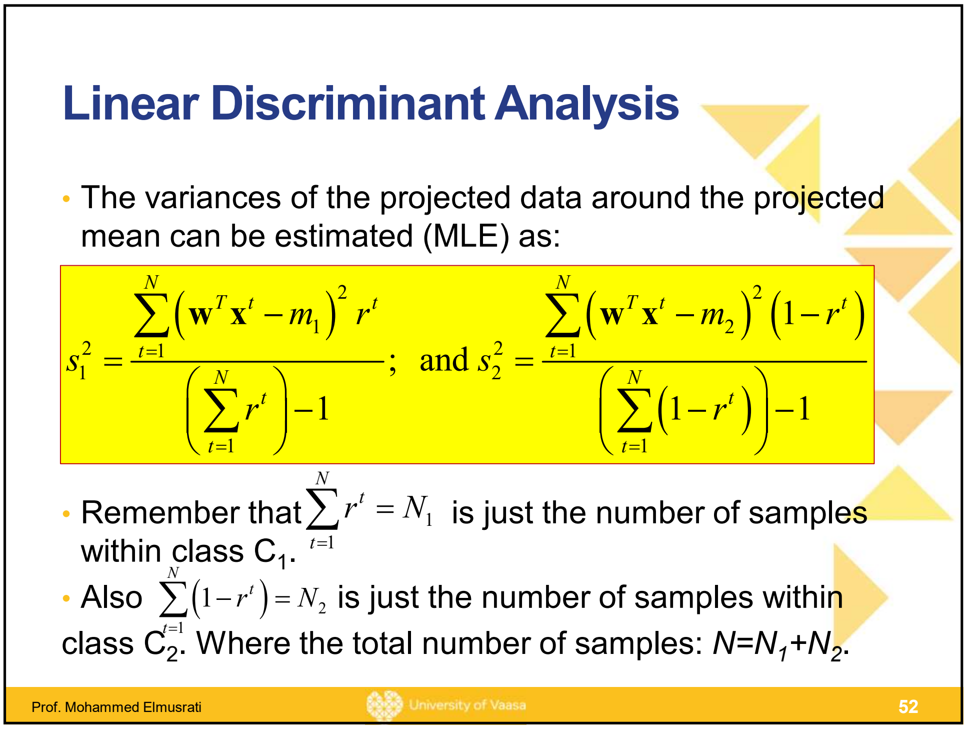
- Given the samples data $X=\{\boldsymbol{x}^t, r^t\}$, such that $r^t=1$ if $\boldsymbol{x}^t \in C_1$, and $r^t=0$ if $\boldsymbol{x}^t \in C_2$, then the MLE estimate of the class average is given by:

$$\mathbf{m}_1 = \frac{\sum_{t=1}^{N} \mathbf{x}^t r^t}{\sum_{t=1}^{N} r^t}; \text{ and } \mathbf{m}_2 = \frac{\sum_{t=1}^{N} \mathbf{x}^t \left(1-r^t\right)}{\sum_{t=1}^{N} \left(1-r^t\right)}$$

- The projection of the mean vector over the weight **w**, we obtain:

$$m_1 = \mathbf{w}^T \mathbf{m}_1; \text{ and } m_2 = \mathbf{w}^T \mathbf{m}_2$$

51



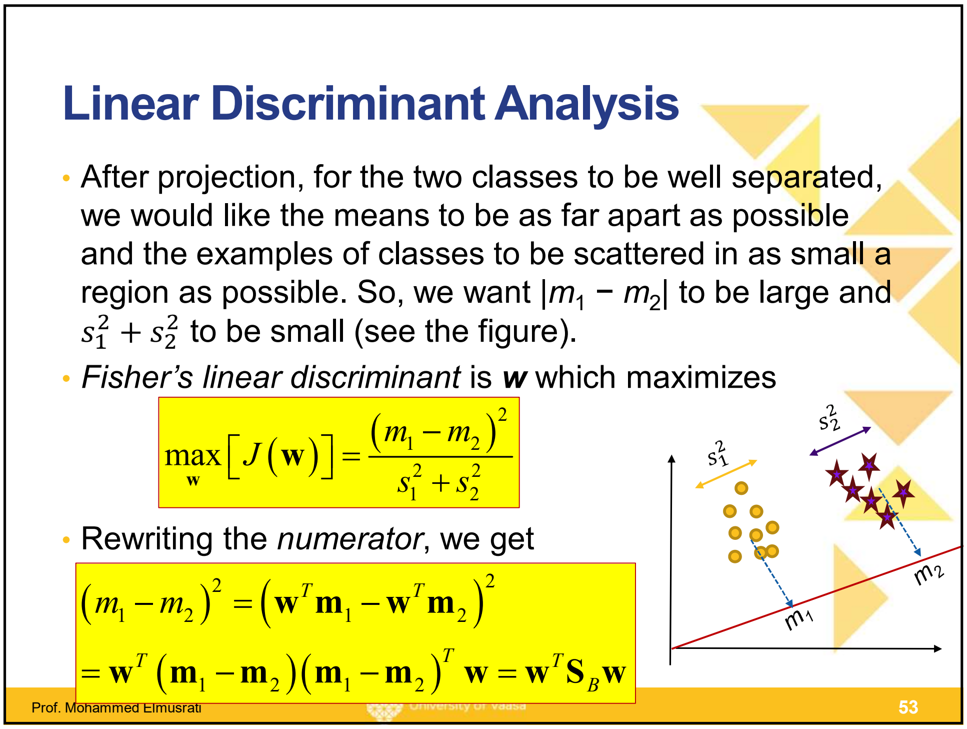
Linear Discriminant Analysis

- The variances of the projected data around the projected mean can be estimated (MLE) as:

$$s_1^2 = \frac{\sum_{t=1}^{N} \left(\mathbf{w}^T \mathbf{x}^t - m_1\right)^2 r^t}{\left(\sum_{t=1}^{N} r^t\right) - 1}; \text{ and } s_2^2 = \frac{\sum_{t=1}^{N} \left(\mathbf{w}^T \mathbf{x}^t - m_2\right)^2 \left(1-r^t\right)}{\left(\sum_{t=1}^{N} \left(1-r^t\right)\right) - 1}$$

- Remember that $\sum_{t=1}^{N} r^t = N_1$ is just the number of samples within class C$_1$.
- Also $\sum_{t=1}^{N} \left(1-r^t\right) = N_2$ is just the number of samples within class C$_2$. Where the total number of samples: $N=N_1+N_2$.

52



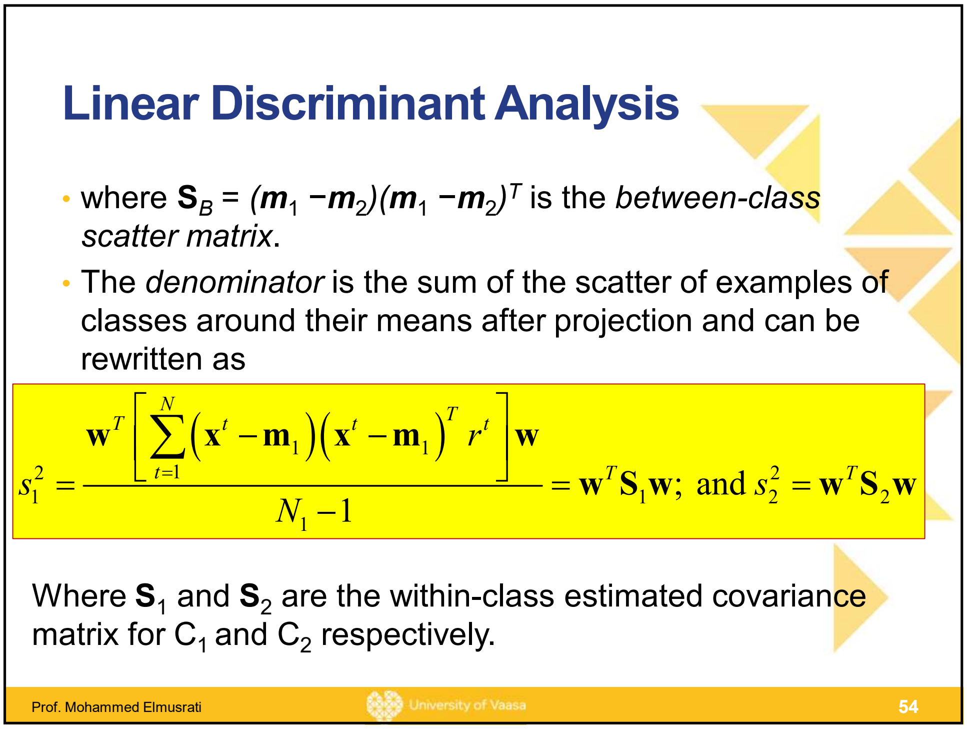
Linear Discriminant Analysis

- After projection, for the two classes to be well separated, we would like the means to be as far apart as possible and the examples of classes to be scattered in as small a region as possible. So, we want $|m_1 - m_2|$ to be large and $s_1^2 + s_2^2$ to be small (see the figure).
- *Fisher's linear discriminant* is ***w*** which maximizes

$$\max_{\mathbf{w}}\left[J(\mathbf{w})\right]=\frac{(m_1-m_2)^2}{s_1^2+s_2^2}$$

- Rewriting the *numerator*, we get

$$(m_1-m_2)^2=\left(\mathbf{w}^T\mathbf{m}_1-\mathbf{w}^T\mathbf{m}_2\right)^2$$
$$=\mathbf{w}^T(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{w}=\mathbf{w}^T\mathbf{S}_B\mathbf{w}$$

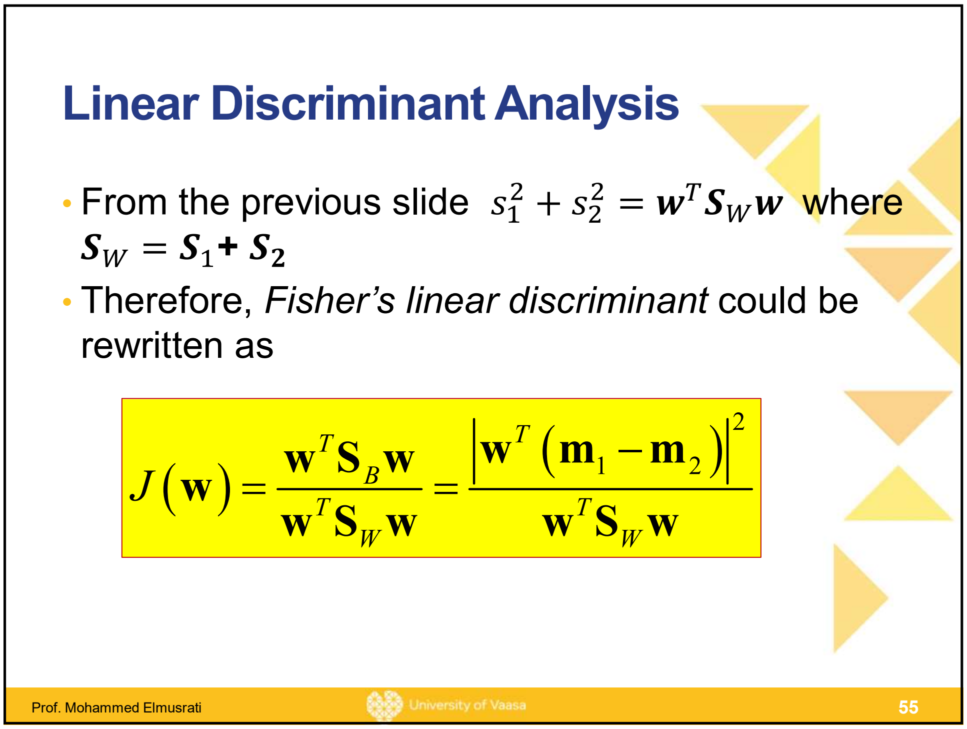


Linear Discriminant Analysis

- where $\mathbf{S}_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$ is the *between-class scatter matrix*.
- The *denominator* is the sum of the scatter of examples of classes around their means after projection and can be rewritten as

$$s_1^2=\frac{\mathbf{w}^T\left[\sum_{t=1}^{N}\left(\mathbf{x}^t-\mathbf{m}_1\right)\left(\mathbf{x}^t-\mathbf{m}_1\right)^T r^t\right]\mathbf{w}}{N_1-1}=\mathbf{w}^T\mathbf{S}_1\mathbf{w};\ \text{and}\ s_2^2=\mathbf{w}^T\mathbf{S}_2\mathbf{w}$$

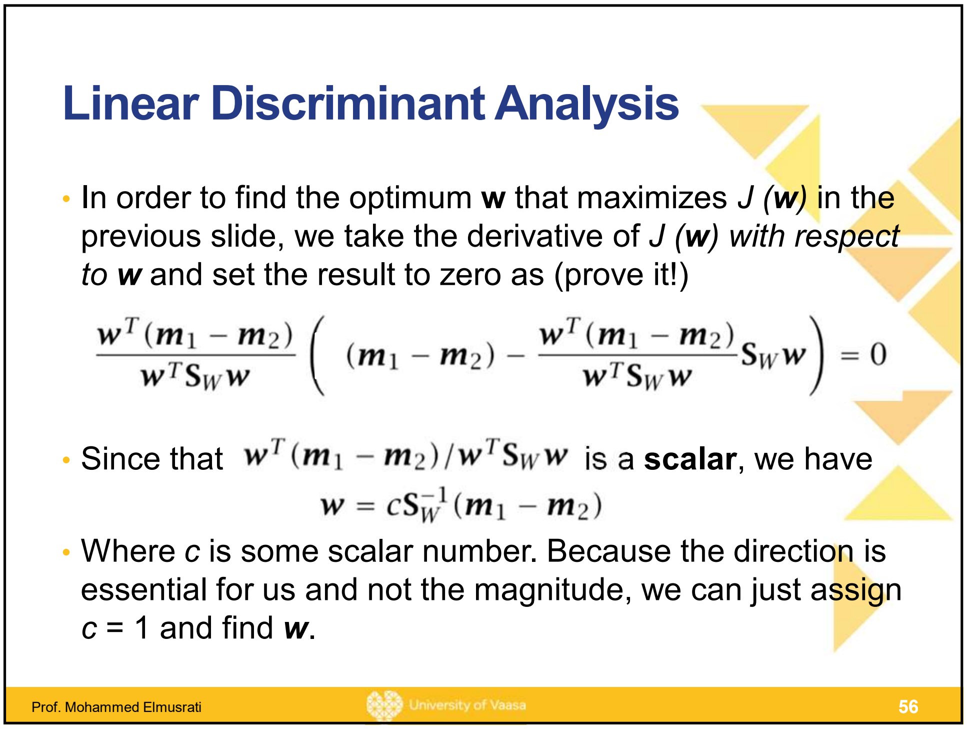
Where $\mathbf{S}_1$ and $\mathbf{S}_2$ are the within-class estimated covariance matrix for C_1 and C_2 respectively.



Linear Discriminant Analysis

- From the previous slide $s_1^2 + s_2^2 = \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}$ where $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$
- Therefore, *Fisher's linear discriminant* could be rewritten as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{\left|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$



Linear Discriminant Analysis

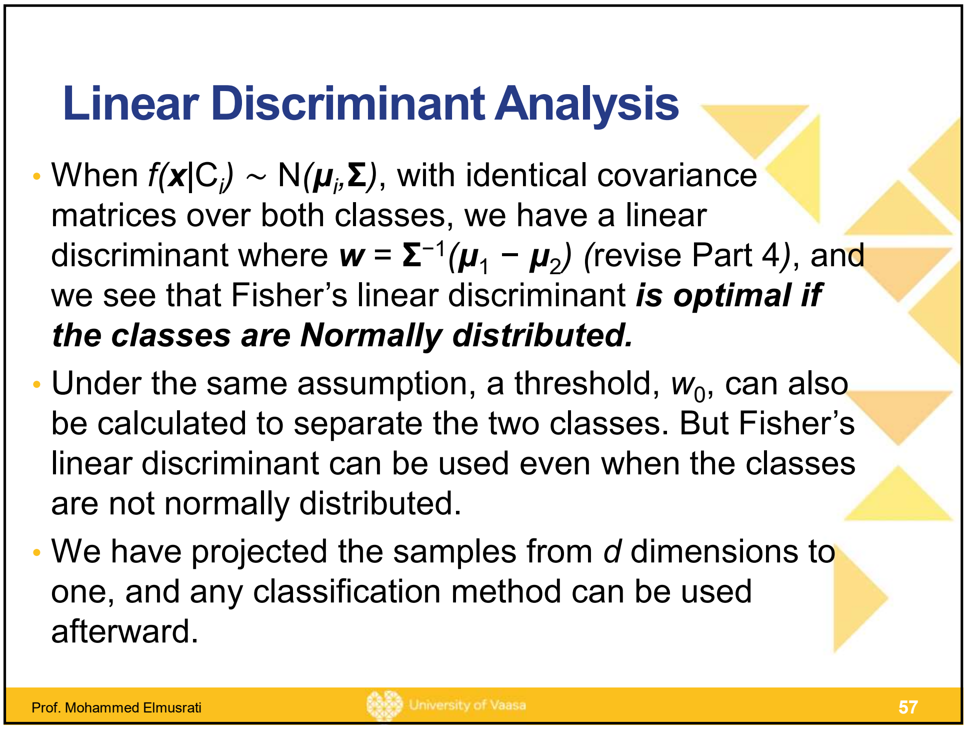
- In order to find the optimum **w** that maximizes *J* (***w***) in the previous slide, we take the derivative of *J* (***w***) *with respect to **w*** and set the result to zero as (prove it!)

$$\frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T \mathrm{S}_W \boldsymbol{w}}\left((\boldsymbol{m}_1 - \boldsymbol{m}_2) - \frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T \mathrm{S}_W \boldsymbol{w}} \mathrm{S}_W \boldsymbol{w}\right) = 0$$

- Since that $\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)/\boldsymbol{w}^T \mathrm{S}_W \boldsymbol{w}$ is a **scalar**, we have

$$\boldsymbol{w} = c\mathrm{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

- Where *c* is some scalar number. Because the direction is essential for us and not the magnitude, we can just assign *c* = 1 and find ***w***.

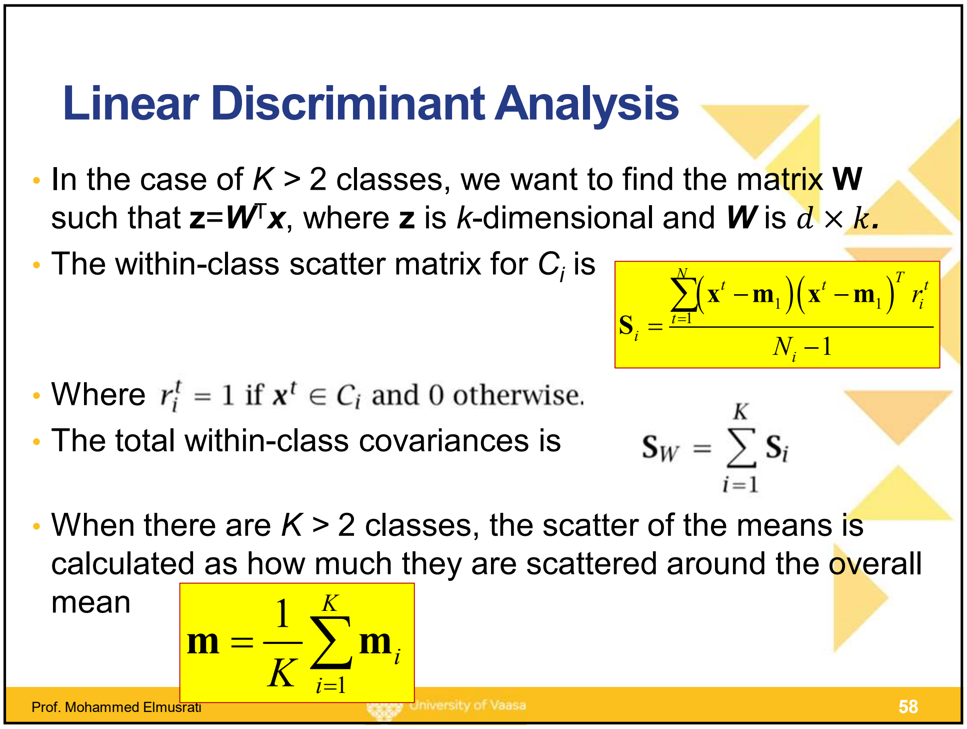


Linear Discriminant Analysis

- When $f(\mathbf{x}|C_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, with identical covariance matrices over both classes, we have a linear discriminant where $\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ (revise Part 4), and we see that Fisher's linear discriminant ***is optimal if the classes are Normally distributed.***
- Under the same assumption, a threshold, w_0, can also be calculated to separate the two classes. But Fisher's linear discriminant can be used even when the classes are not normally distributed.
- We have projected the samples from *d* dimensions to one, and any classification method can be used afterward.

Prof. Mohammed Elmusrati University of Vaasa 57

57



Linear Discriminant Analysis

- In the case of $K > 2$ classes, we want to find the matrix **W** such that $\mathbf{z}=\boldsymbol{W}^T\boldsymbol{x}$, where **z** is *k*-dimensional and ***W*** is $d \times k$.
- The within-class scatter matrix for C_i is

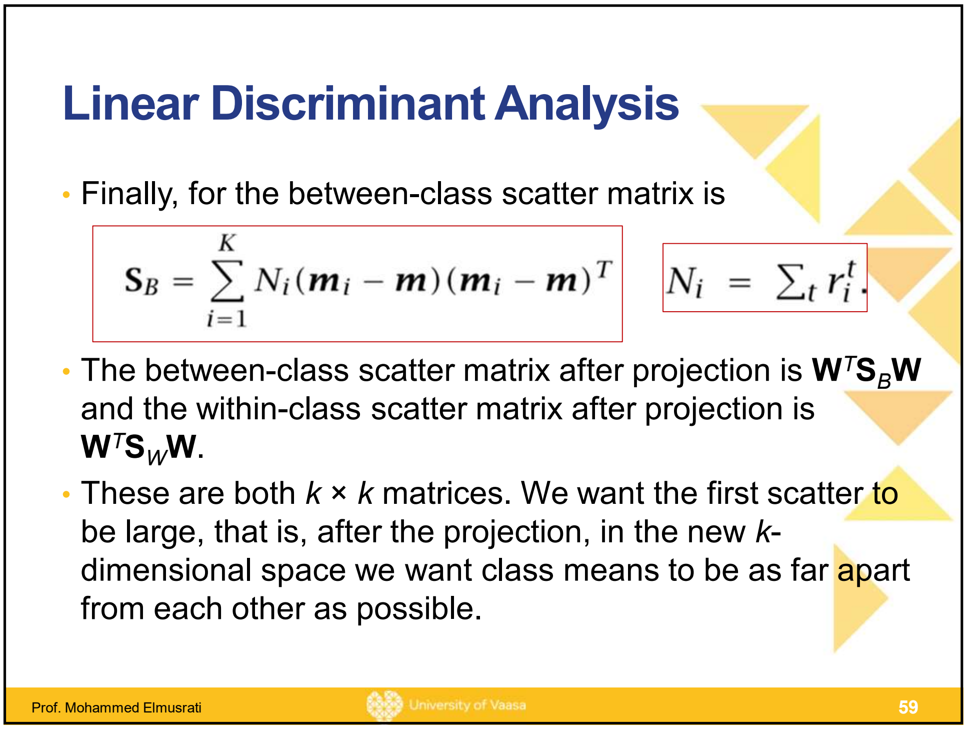
$$\mathbf{S}_i = \frac{\sum_{t=1}^{N} (\mathbf{x}^t - \mathbf{m}_1)(\mathbf{x}^t - \mathbf{m}_1)^T r_i^t}{N_i - 1}$$

- Where $r_i^t = 1$ if $\mathbf{x}^t \in C_i$ and 0 otherwise.
- The total within-class covariances is $\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i$
- When there are $K > 2$ classes, the scatter of the means is calculated as how much they are scattered around the overall mean

$$\mathbf{m} = \frac{1}{K} \sum_{i=1}^{K} \mathbf{m}_i$$

Prof. Mohammed Elmusrati University of Vaasa 58

58



Linear Discriminant Analysis

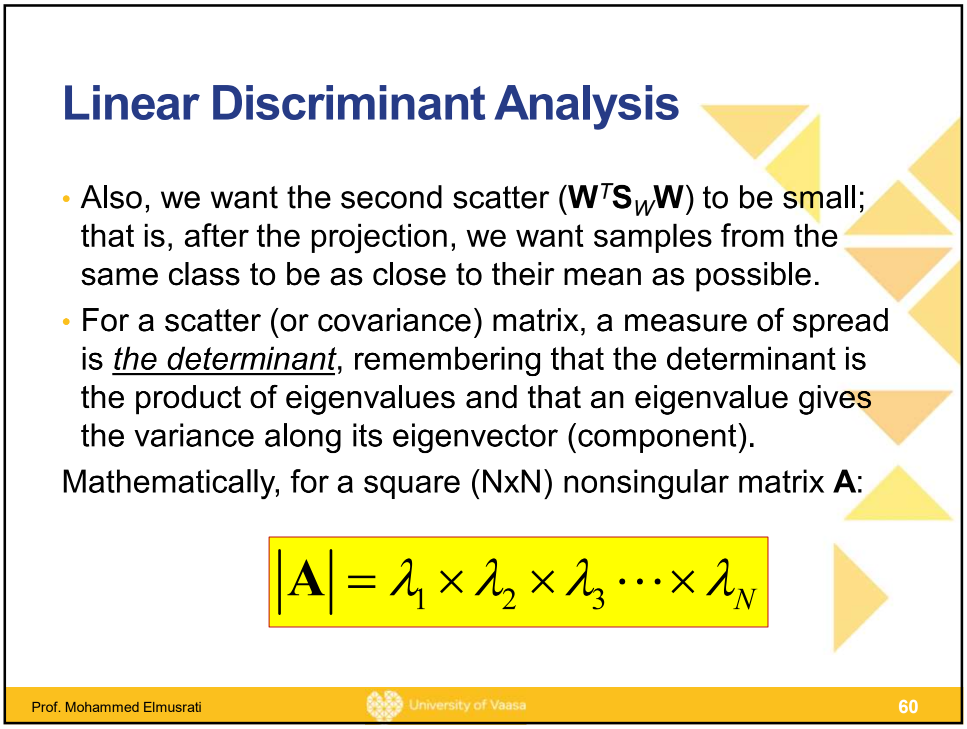
- Finally, for the between-class scatter matrix is

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

$$N_i = \sum_t r_i^t .$$

- The between-class scatter matrix after projection is $\mathbf{W}^T\mathbf{S}_B\mathbf{W}$ and the within-class scatter matrix after projection is $\mathbf{W}^T\mathbf{S}_W\mathbf{W}$.
- These are both *k* × *k* matrices. We want the first scatter to be large, that is, after the projection, in the new *k*-dimensional space we want class means to be as far apart from each other as possible.

Prof. Mohammed Elmusrati University of Vaasa 59



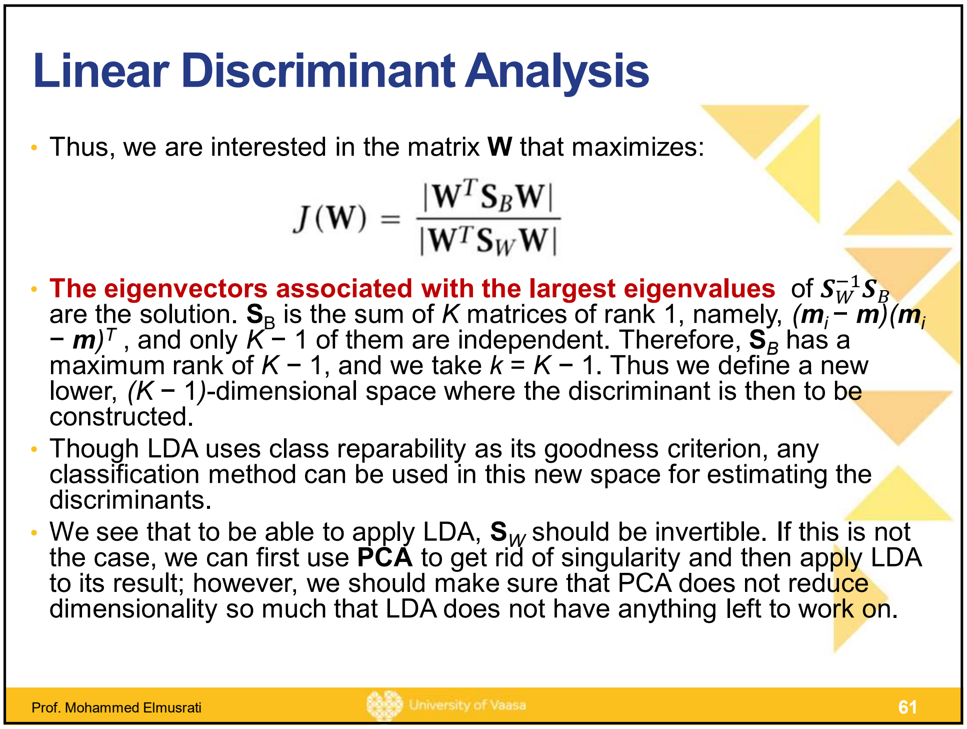
Linear Discriminant Analysis

- Also, we want the second scatter ($\mathbf{W}^T\mathbf{S}_W\mathbf{W}$) to be small; that is, after the projection, we want samples from the same class to be as close to their mean as possible.
- For a scatter (or covariance) matrix, a measure of spread is *the determinant*, remembering that the determinant is the product of eigenvalues and that an eigenvalue gives the variance along its eigenvector (component).

Mathematically, for a square (NxN) nonsingular matrix **A**:

$$|\mathbf{A}| = \lambda_1 \times \lambda_2 \times \lambda_3 \cdots \times \lambda_N$$

Prof. Mohammed Elmusrati University of Vaasa 60

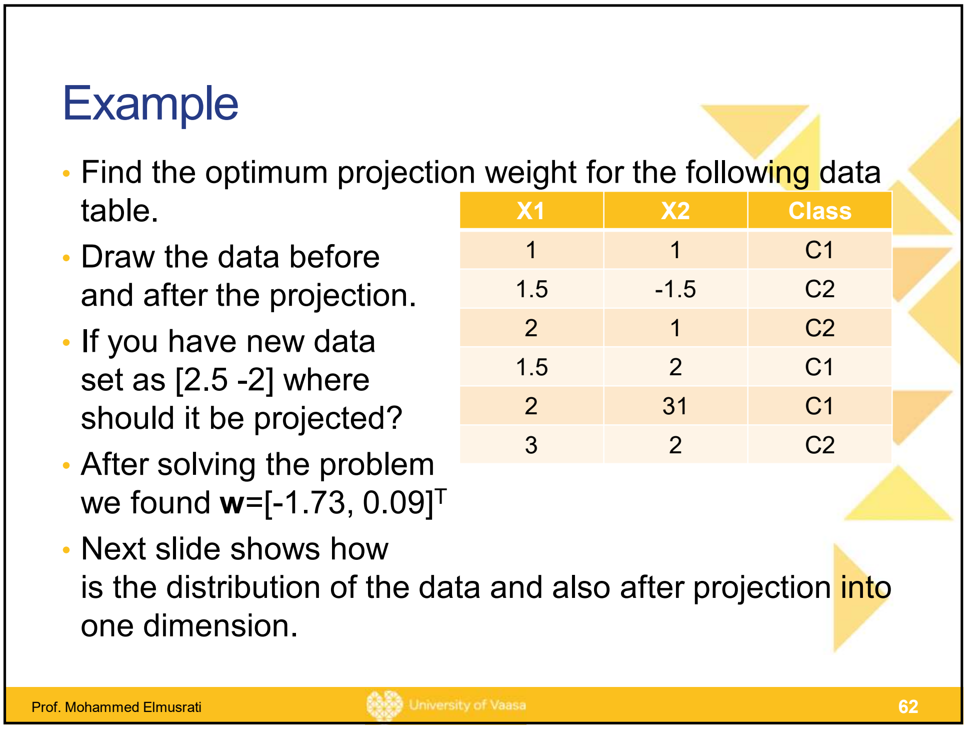


Linear Discriminant Analysis

- Thus, we are interested in the matrix **W** that maximizes:

$$J(\mathbf{W}) = \frac{|\mathbf{W}^T \mathbf{S}_B \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_W \mathbf{W}|}$$

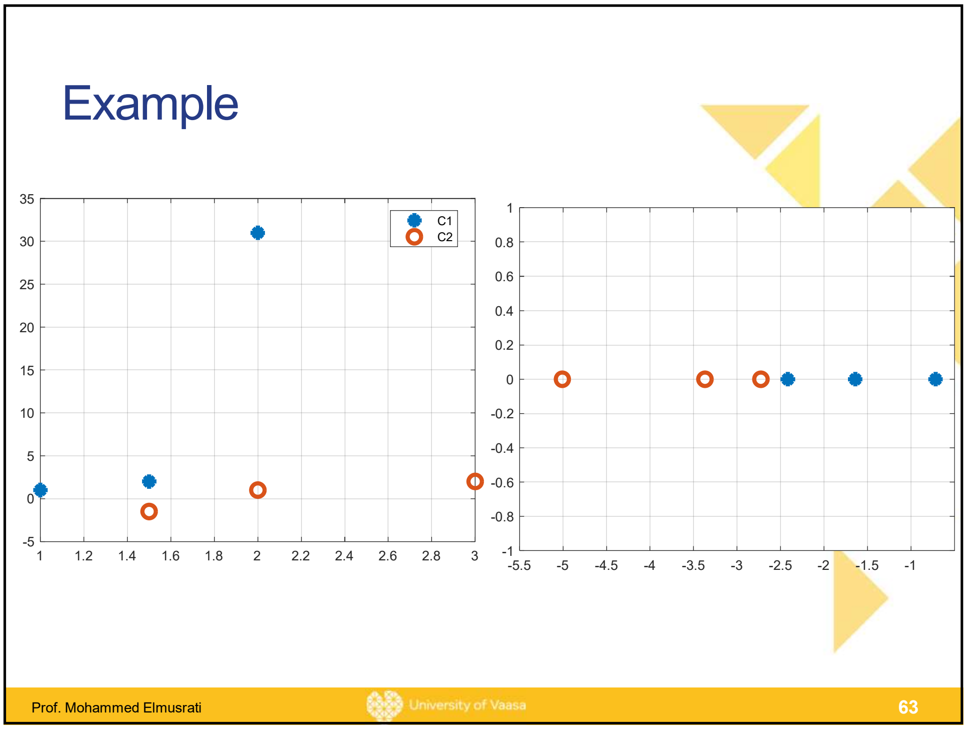
- **The eigenvectors associated with the largest eigenvalues** of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ are the solution. $\mathbf{S}_B$ is the sum of *K* matrices of rank 1, namely, $(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$, and only *K* − 1 of them are independent. Therefore, $\mathbf{S}_B$ has a maximum rank of *K* − 1, and we take *k* = *K* − 1. Thus we define a new lower, *(K* − 1*)*-dimensional space where the discriminant is then to be constructed.
- Though LDA uses class reparability as its goodness criterion, any classification method can be used in this new space for estimating the discriminants.
- We see that to be able to apply LDA, $\mathbf{S}_W$ should be invertible. If this is not the case, we can first use **PCA** to get rid of singularity and then apply LDA to its result; however, we should make sure that PCA does not reduce dimensionality so much that LDA does not have anything left to work on.



Example

- Find the optimum projection weight for the following data table.
- Draw the data before and after the projection.
- If you have new data set as [2.5 -2] where should it be projected?
- After solving the problem we found **w**=[-1.73, 0.09]^T
- Next slide shows how is the distribution of the data and also after projection into one dimension.

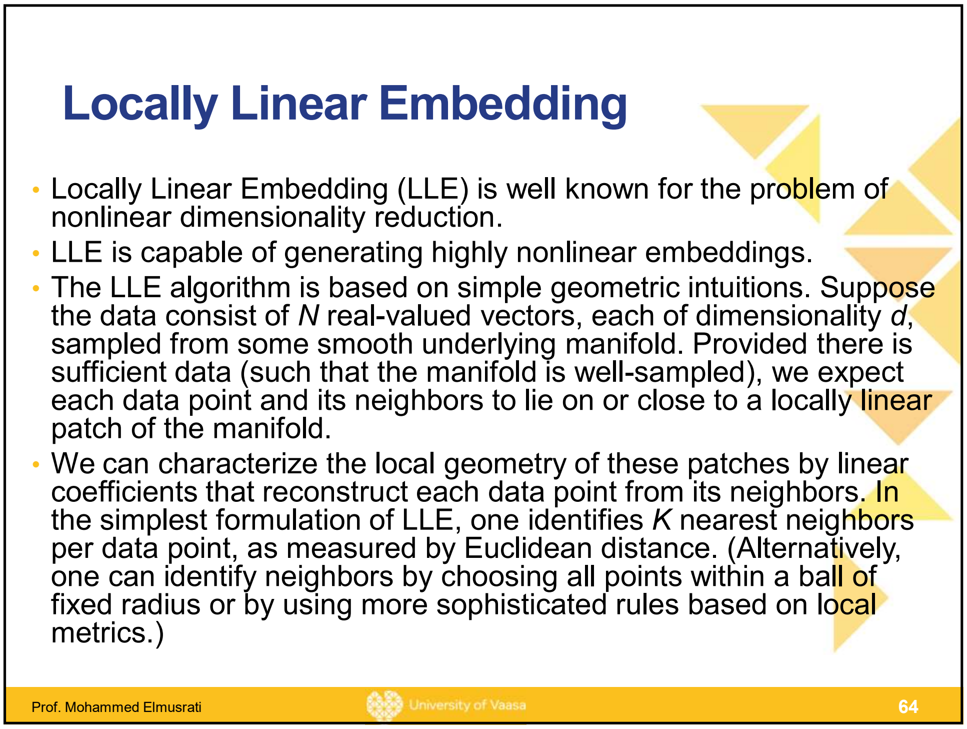
X1	X2	Class
1	1	C1
1.5	-1.5	C2
2	1	C2
1.5	2	C1
2	31	C1
3	2	C2



Example

Prof. Mohammed Elmusrati University of Vaasa 63

63

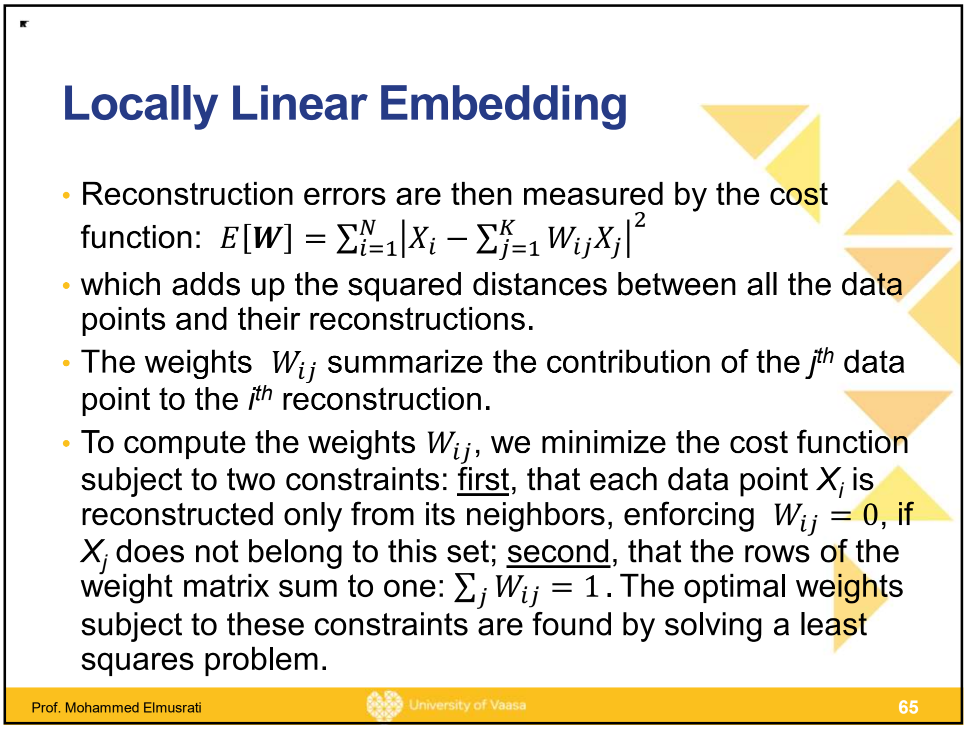


Locally Linear Embedding

- Locally Linear Embedding (LLE) is well known for the problem of nonlinear dimensionality reduction.
- LLE is capable of generating highly nonlinear embeddings.
- The LLE algorithm is based on simple geometric intuitions. Suppose the data consist of *N* real-valued vectors, each of dimensionality *d*, sampled from some smooth underlying manifold. Provided there is sufficient data (such that the manifold is well-sampled), we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold.
- We can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. In the simplest formulation of LLE, one identifies *K* nearest neighbors per data point, as measured by Euclidean distance. (Alternatively, one can identify neighbors by choosing all points within a ball of fixed radius or by using more sophisticated rules based on local metrics.)

Prof. Mohammed Elmusrati University of Vaasa 64

64

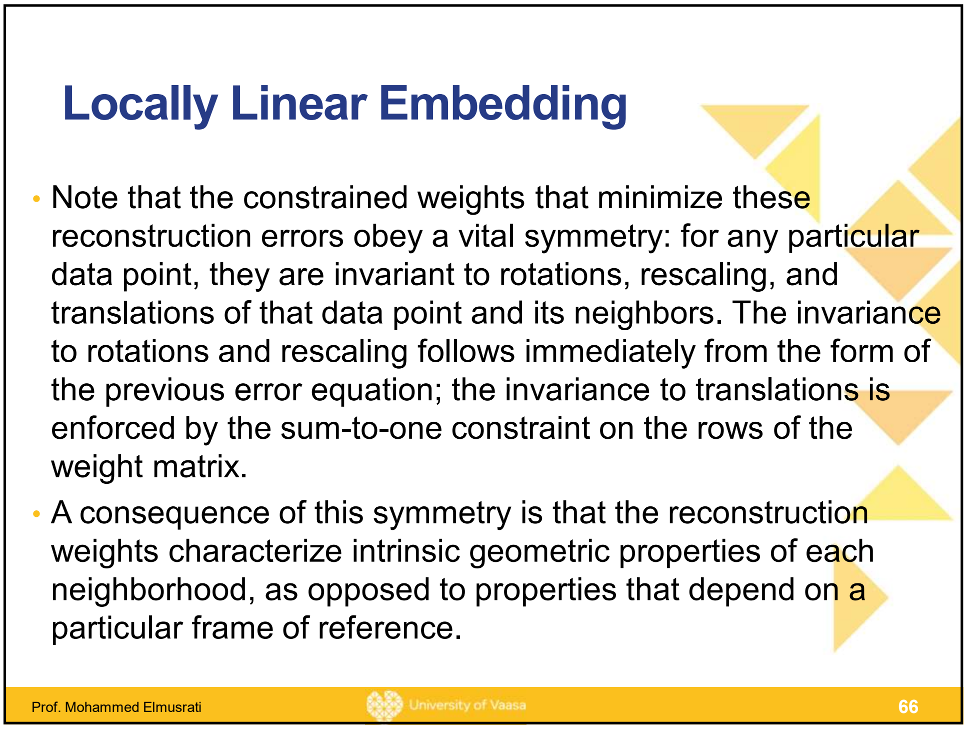


Locally Linear Embedding

- Reconstruction errors are then measured by the cost function: $E[\boldsymbol{W}] = \sum_{i=1}^{N} \left| X_i - \sum_{j=1}^{K} W_{ij} X_j \right|^2$
- which adds up the squared distances between all the data points and their reconstructions.
- The weights W_{ij} summarize the contribution of the *j*th data point to the *i*th reconstruction.
- To compute the weights W_{ij}, we minimize the cost function subject to two constraints: first, that each data point X_i is reconstructed only from its neighbors, enforcing $W_{ij} = 0$, if X_j does not belong to this set; second, that the rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$. The optimal weights subject to these constraints are found by solving a least squares problem.

Prof. Mohammed Elmusrati University of Vaasa 65

65

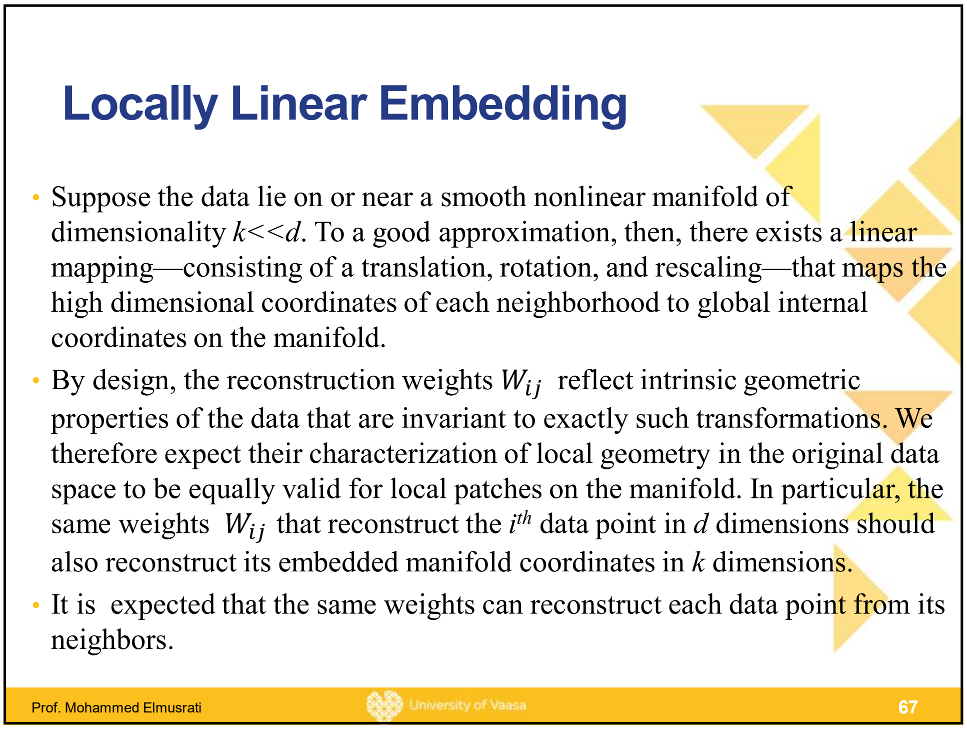


Locally Linear Embedding

- Note that the constrained weights that minimize these reconstruction errors obey a vital symmetry: for any particular data point, they are invariant to rotations, rescaling, and translations of that data point and its neighbors. The invariance to rotations and rescaling follows immediately from the form of the previous error equation; the invariance to translations is enforced by the sum-to-one constraint on the rows of the weight matrix.
- A consequence of this symmetry is that the reconstruction weights characterize intrinsic geometric properties of each neighborhood, as opposed to properties that depend on a particular frame of reference.

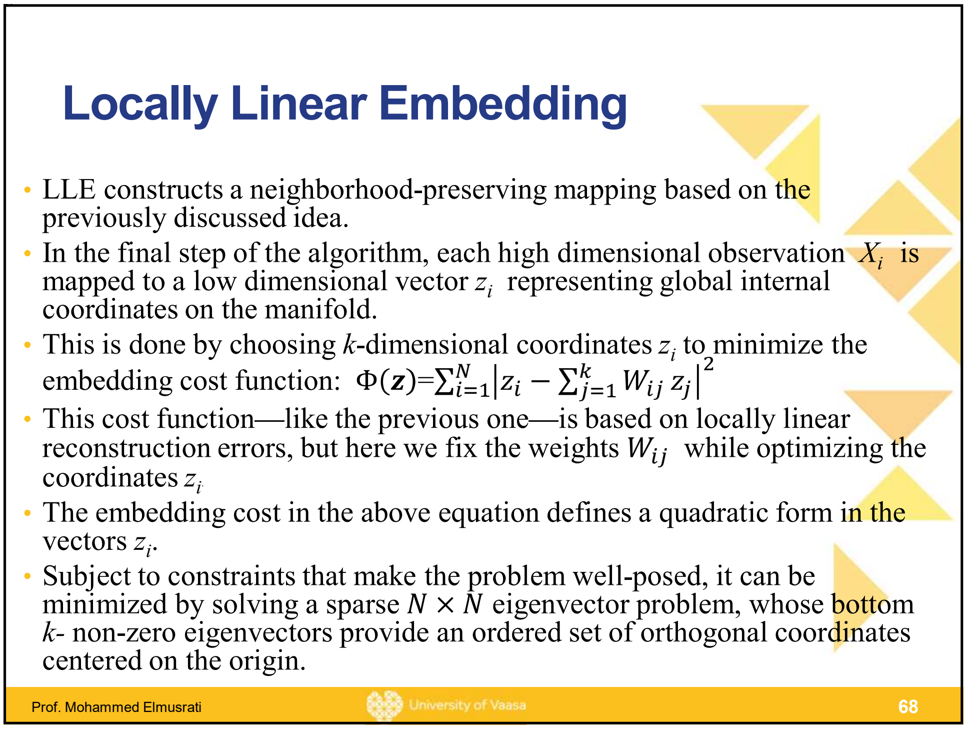
Prof. Mohammed Elmusrati University of Vaasa 66

66



Locally Linear Embedding

- Suppose the data lie on or near a smooth nonlinear manifold of dimensionality k<<d. To a good approximation, then, there exists a linear mapping—consisting of a translation, rotation, and rescaling—that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold.
- By design, the reconstruction weights W_{ij} reflect intrinsic geometric properties of the data that are invariant to exactly such transformations. We therefore expect their characterization of local geometry in the original data space to be equally valid for local patches on the manifold. In particular, the same weights W_{ij} that reconstruct the i^{th} data point in d dimensions should also reconstruct its embedded manifold coordinates in k dimensions.
- It is expected that the same weights can reconstruct each data point from its neighbors.



Locally Linear Embedding

- LLE constructs a neighborhood-preserving mapping based on the previously discussed idea.
- In the final step of the algorithm, each high dimensional observation X_i is mapped to a low dimensional vector z_i representing global internal coordinates on the manifold.
- This is done by choosing k-dimensional coordinates z_i to minimize the embedding cost function: $\Phi(\boldsymbol{z})=\sum_{i=1}^{N}\left|z_i-\sum_{j=1}^{k}W_{ij}\,z_j\right|^2$
- This cost function—like the previous one—is based on locally linear reconstruction errors, but here we fix the weights W_{ij} while optimizing the coordinates z_i
- The embedding cost in the above equation defines a quadratic form in the vectors z_i.
- Subject to constraints that make the problem well-posed, it can be minimized by solving a sparse $N \times N$ eigenvector problem, whose bottom k- non-zero eigenvectors provide an ordered set of orthogonal coordinates centered on the origin.

Locally Linear Embedding

- Implementation of the algorithm is fairly straightforward, as the algorithm has only one free parameter: the number of neighbors per data point, K.
- Once neighbors are chosen, the optimal weights W_{ij} and coordinates z_i are computed by standard methods in linear algebra.
- Example:
- K=12.

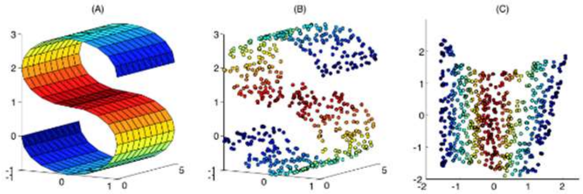
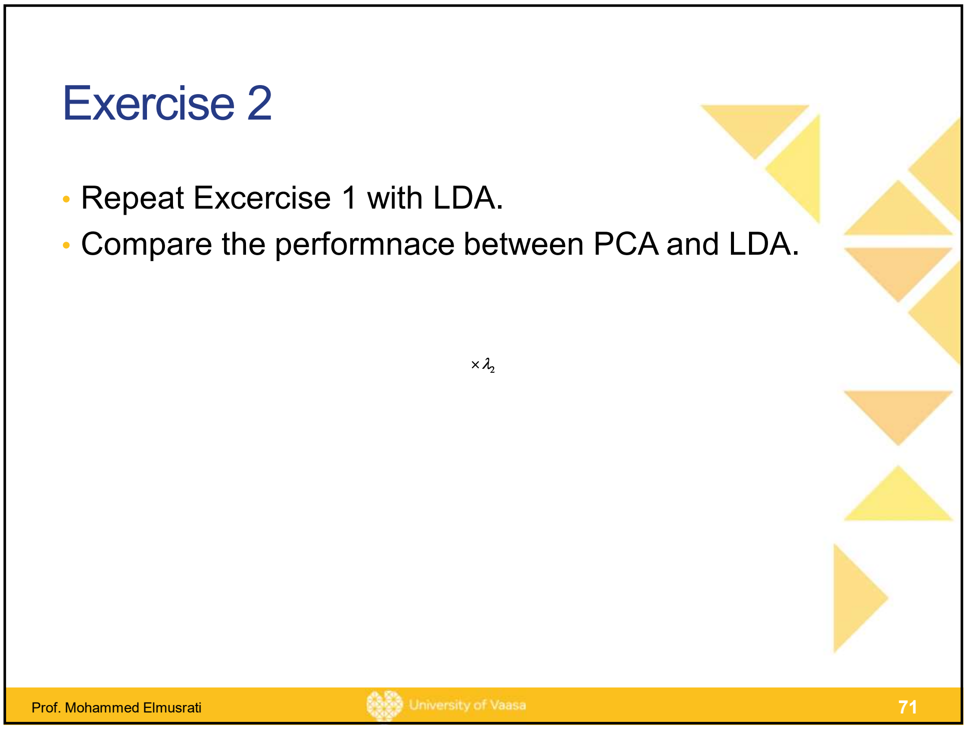


Figure 1: The problem of nonlinear dimensionality reduction, as illustrated for three dimensional data (B) sampled from a two dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The shading in (C) illustrates the neighborhood-preserving mapping discovered by LLE.

Exercise 1

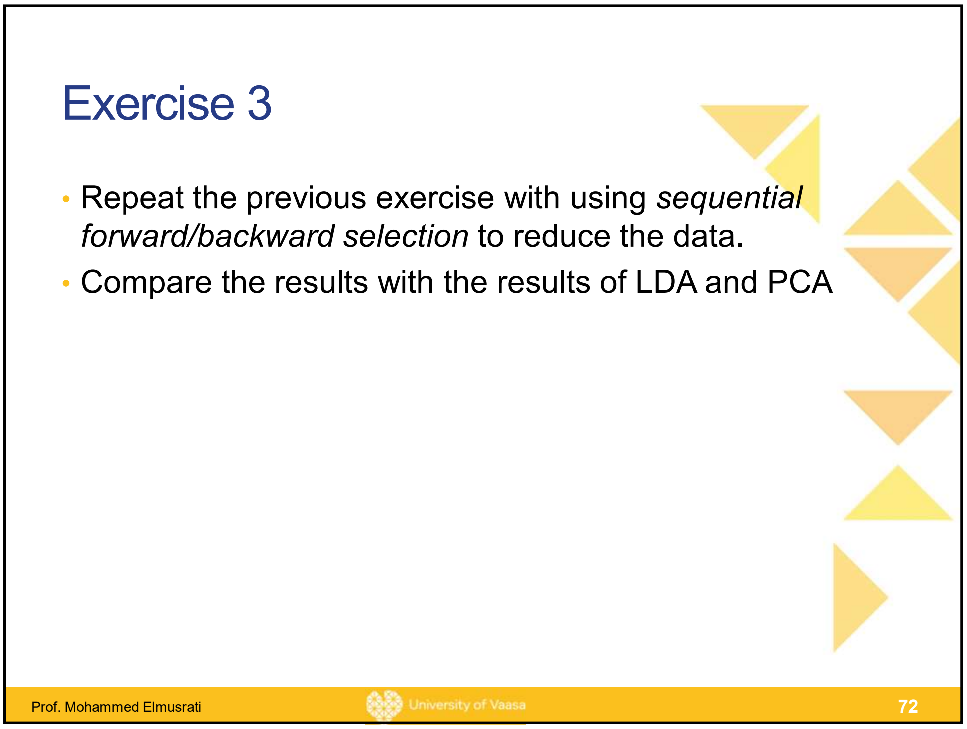
- **Data_Part5_Ex2_Learn** (on Moodle) contains certain data collected represent certain information. It has dimension of 15 and it represents 5 different classes that labelled by Y={0,1,2,3,4}. The length of samples is 200.
- First use **PCA** to minimize the dimension.
- Second build your discrimination function as described in Part 4.
- Third: Validate your code of reduction and classes discrimination by using the second file dataset **Data_Part5_Ex2_test** (on course Moodle).
- It contains 400 data samples.



Exercise 2

- Repeat Excercise 1 with LDA.
- Compare the performnace between PCA and LDA.

$\times \lambda_2$



Exercise 3

- Repeat the previous exercise with using *sequential forward/backward selection* to reduce the data.
- Compare the results with the results of LDA and PCA