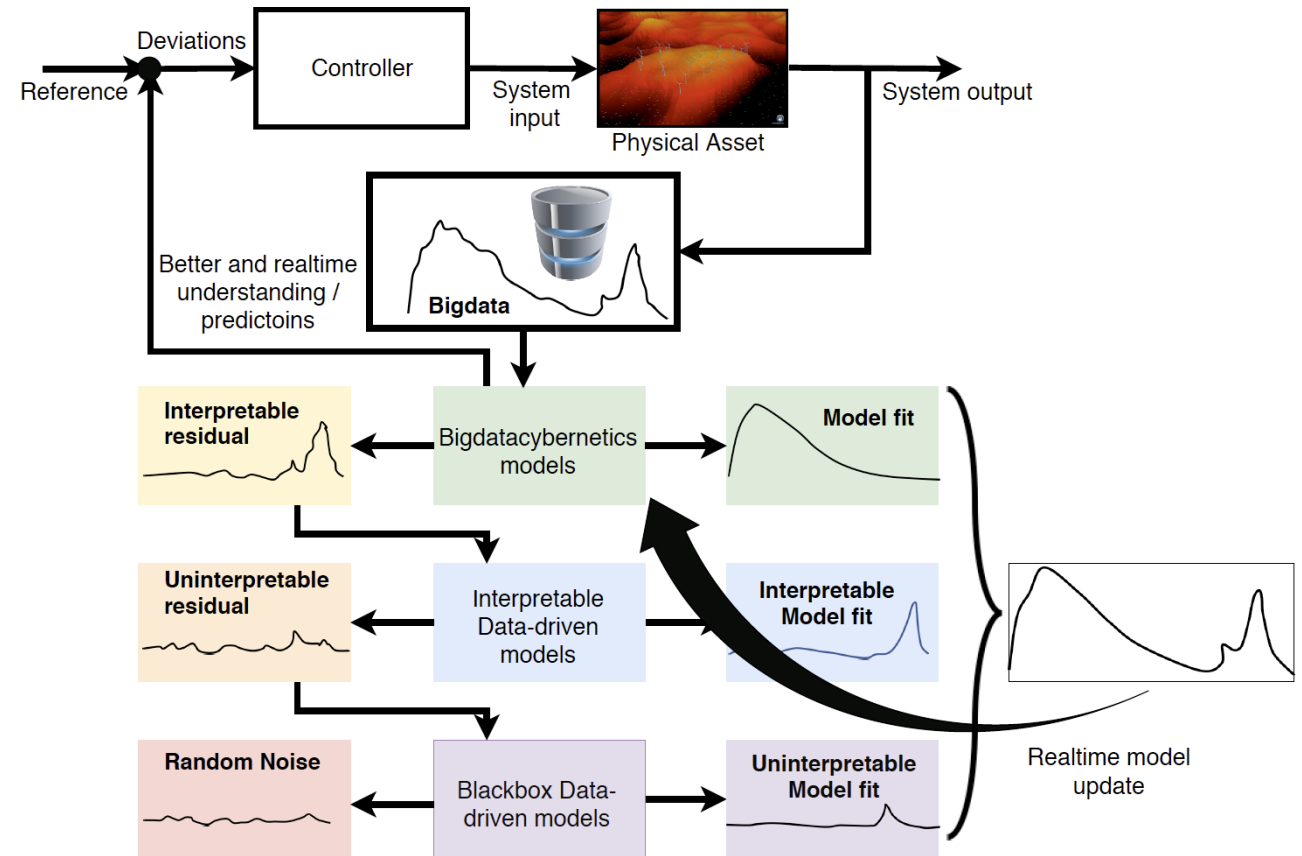


Lecture 5: Multivariate Regression

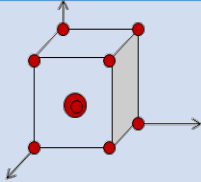
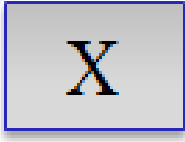
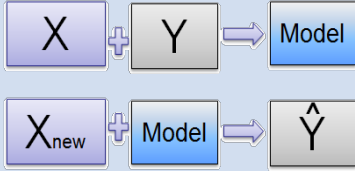
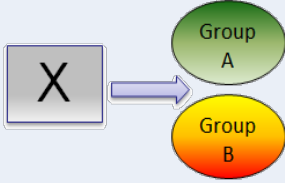
Adil RASHEED and Damiano VARAGNOLO

Sensor measurements

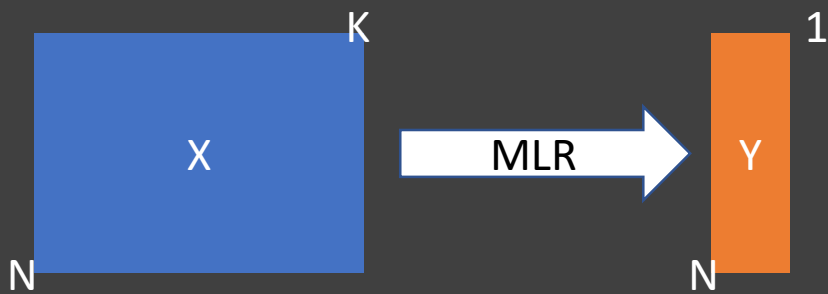
- Direct measurement of the quality of interest not possible or is expensive
- Redundant sensors can be used for robustness and to detect anomalies
- Sensor data can be used for prediction or projections when combined with data analysis techniques



Multivariate Tools and their Purposes

Tool	Symbol	Purpose
Design of Experiments (DoE) Full and Fractional Factorial Designs, Optimisation Designs, Mixture Designs.		Gain maximum information from a minimum of experimental effort
Exploratory Data Analysis (EDA)		Find important sample groups or variable relationships in large data sets.
Regression and Prediction		Predict quality and/or find out which variables that most influence a process
Classification/Discrimination		Determine the state of a process. Identify raw materials. Detect counterfeit samples.

Regression



Ordinary Least Squares (OLS) Regression



Total Least Squares (TLS) Regression



Ridge Regression (RR)



Principal Component Regression (PCR)



Partial Least Squares Regression (PLSR)



Decision Tree (DT) and Random Forest (RF)

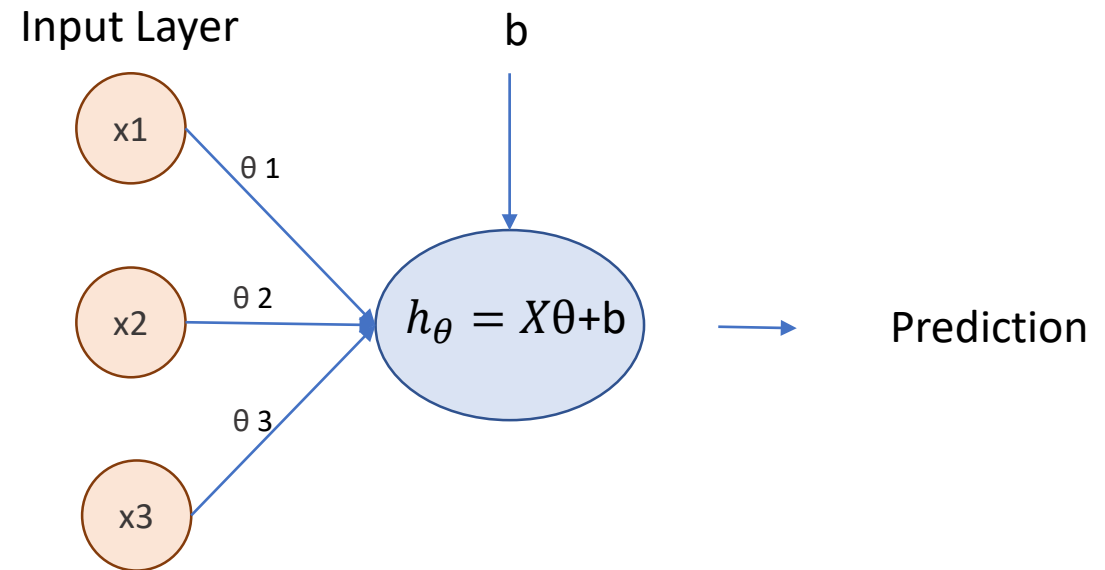


Support Vector Machine (SVM)

Linear Regression

Input feature vectors			Dependent variable
x1	x2	x3	y
...
...
...
...

Easy to interpret 😊

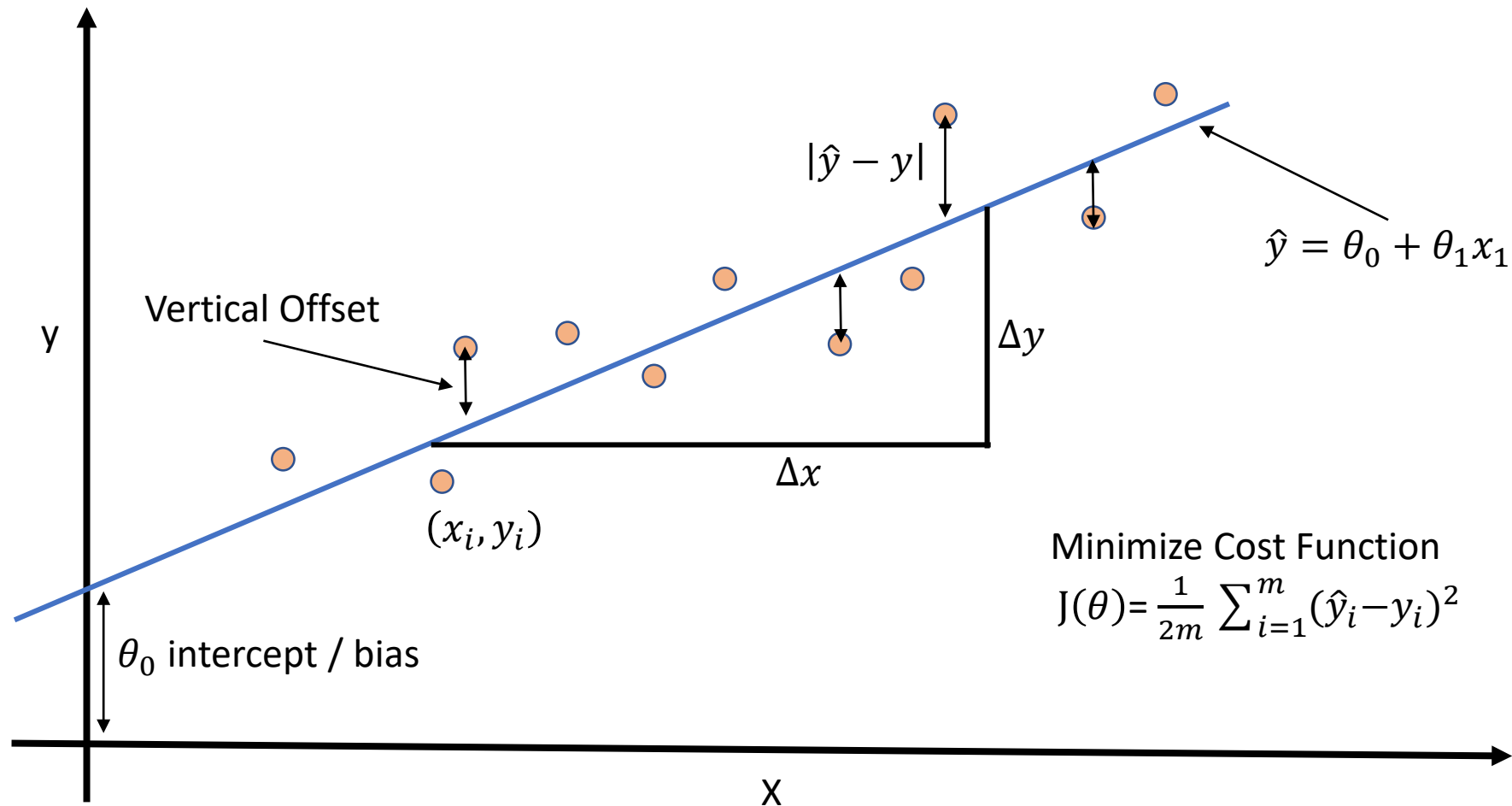


Logistic Regression

- Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier.

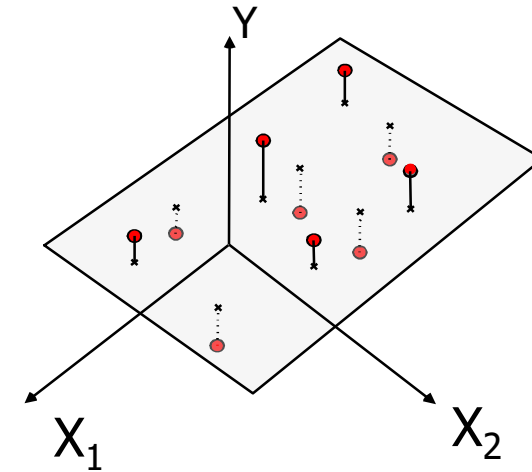
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Multiple Linear / Ordinary Least Square Regression



Regression modeling: Direct Methods

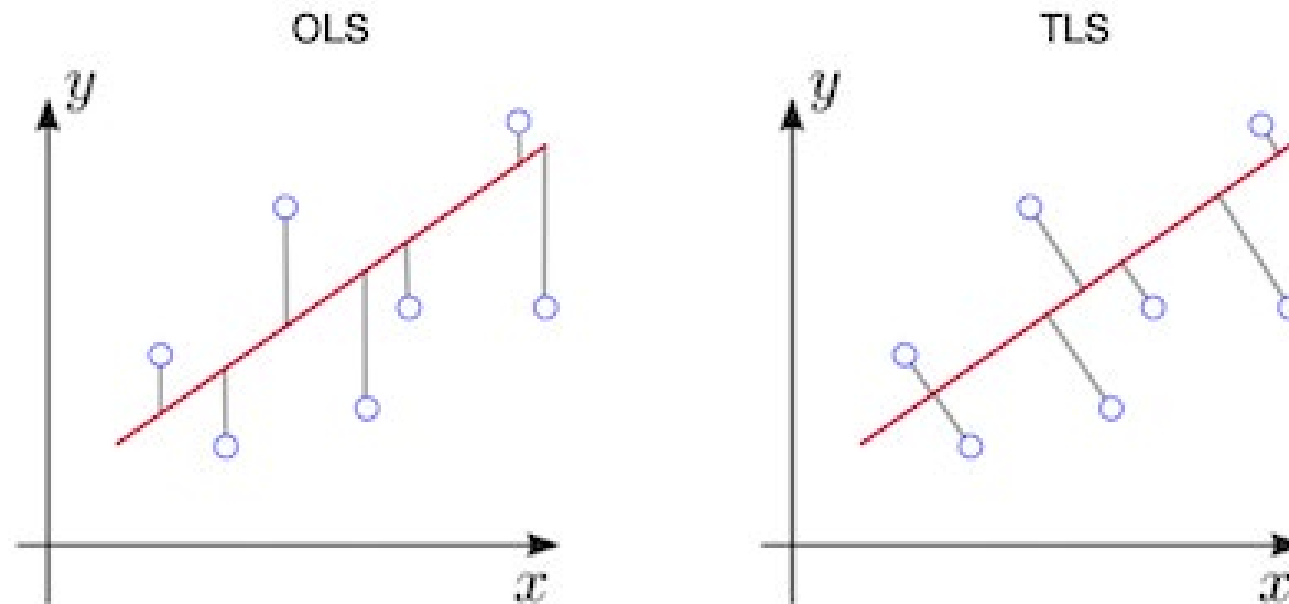
- MLR estimates regression coefficients from the formula
- $y = X\theta + f$
- $\theta = (X^T X)^{-1} X^T Y$ (*full rank required*)
- When the X-Variables are correlated, the MLR solution might be unstable and lead to erroneous interpretation
- The coefficients may not reflect the causality in the system



Least squares criterion

Total Least Regression

- <https://towardsdatascience.com/total-least-squares-in-comparison-with-ols-and-odr-f050ffc1a86a>



Ridge Regression

- Loss function of the Linear Regression is regularized with a constant times the (weight * weight)

Batch Gradient Descent

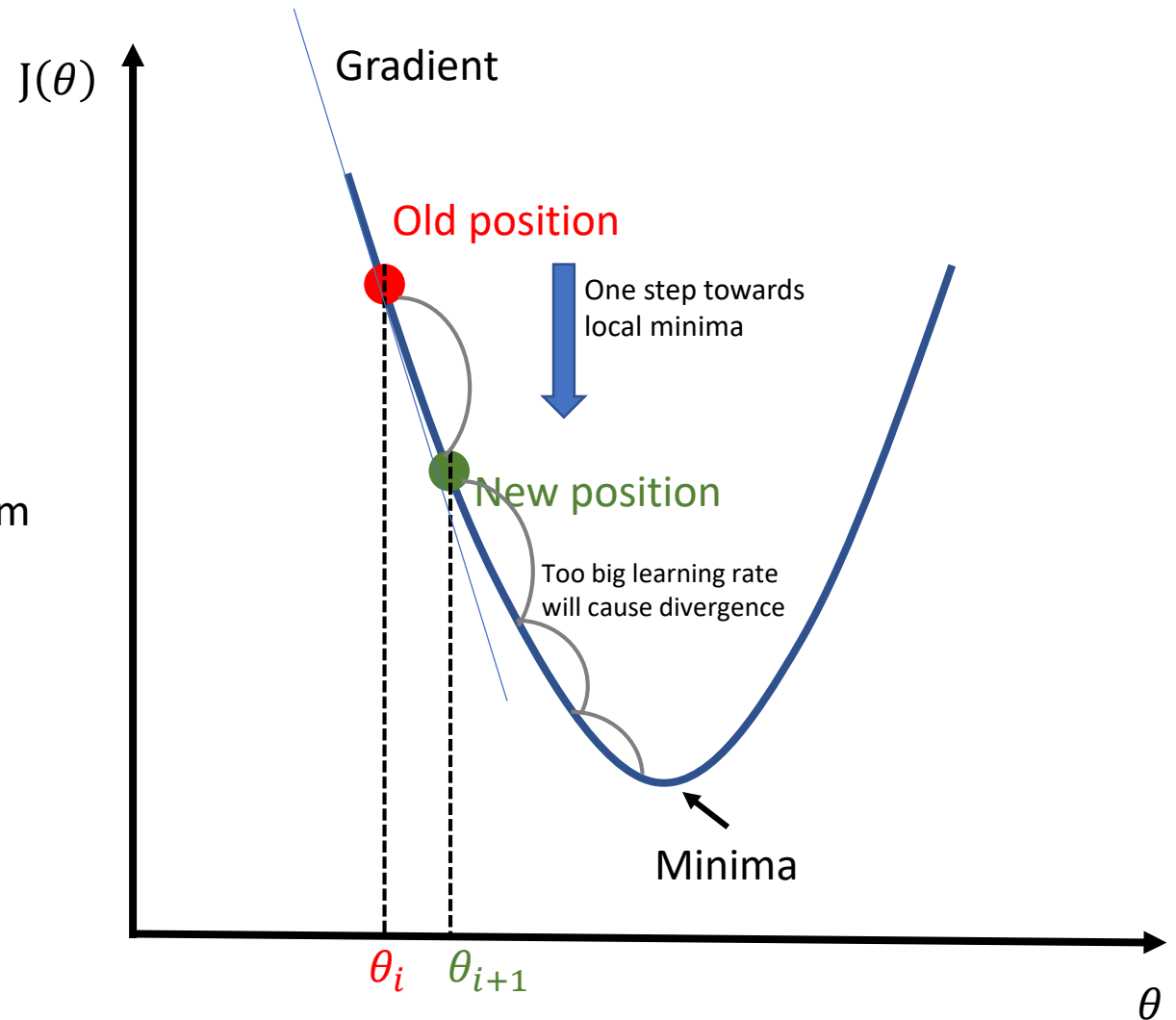
$$\theta_{i+1} = \theta_i - \alpha \nabla_{\theta} J(\theta)$$

New position after the step

Old position before the step

Learning rate

Gradient term



Batch Gradient Descent

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

for every $j = 0, \dots, n$

}

Stochastic Gradient Descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

Repeat {

for $i = 1, \dots, m$ {

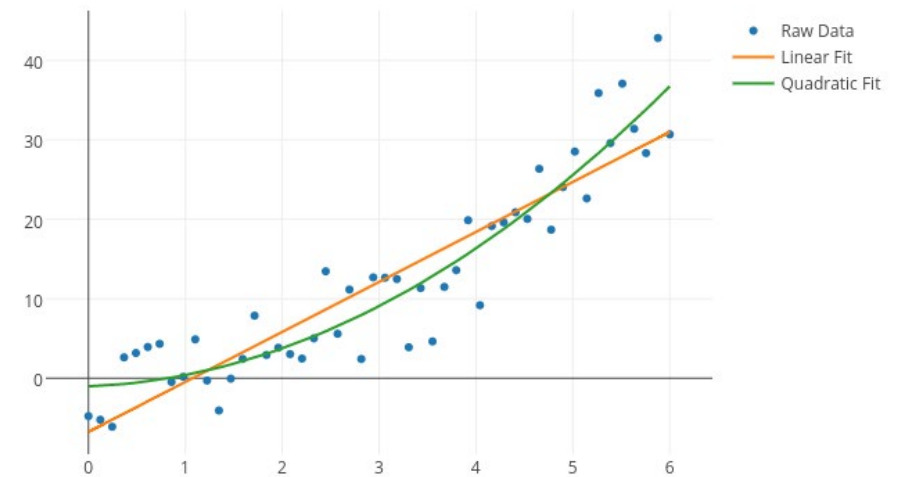
$$\theta_j := \theta_j - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

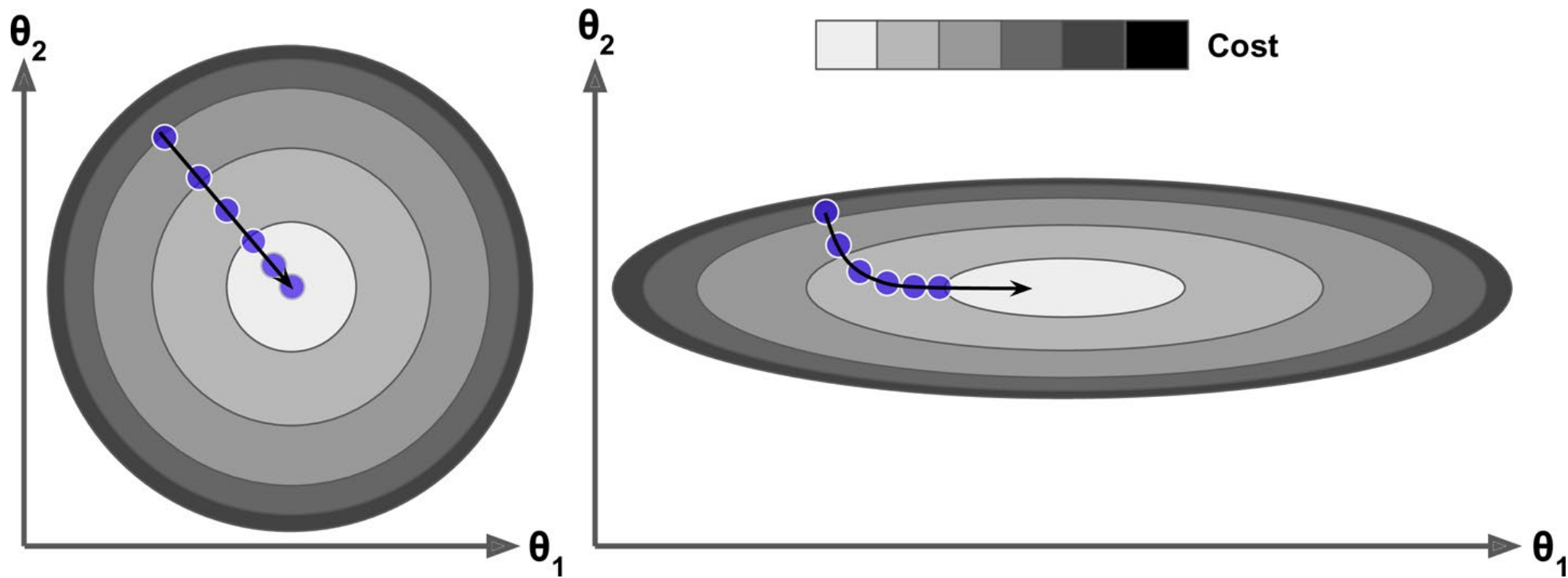
}

}

Linear Regression: + vs -

- Advantages
 - Simplicity
 - Interpretability
 - ...
- Disadvantages
 - Sensitive to outliers
 - Can not handle non-linear data
 - ...





Feature Normalization

Different features can have very different scales:

No. Of rooms 1-6 while area 100-1000m²

Categorical Variables and One-Hot-Encoding

- Male / Female
- Digits 0-9
- Weekdays

Entry for male

	Male	Female
0	0	1
1	1	0

Entry for female

One Hot Encoding for digits 0-9

[illegible]

Interpreting Regression Models

Equations for error measures

- **Residuals** = $y_i - \hat{y}_i$

Computed for each object, i

- **Residual variance** =
$$\frac{\sum_{i=1}^N (y - \hat{y}_i)^2}{N}$$

Overall error on the response variable Y

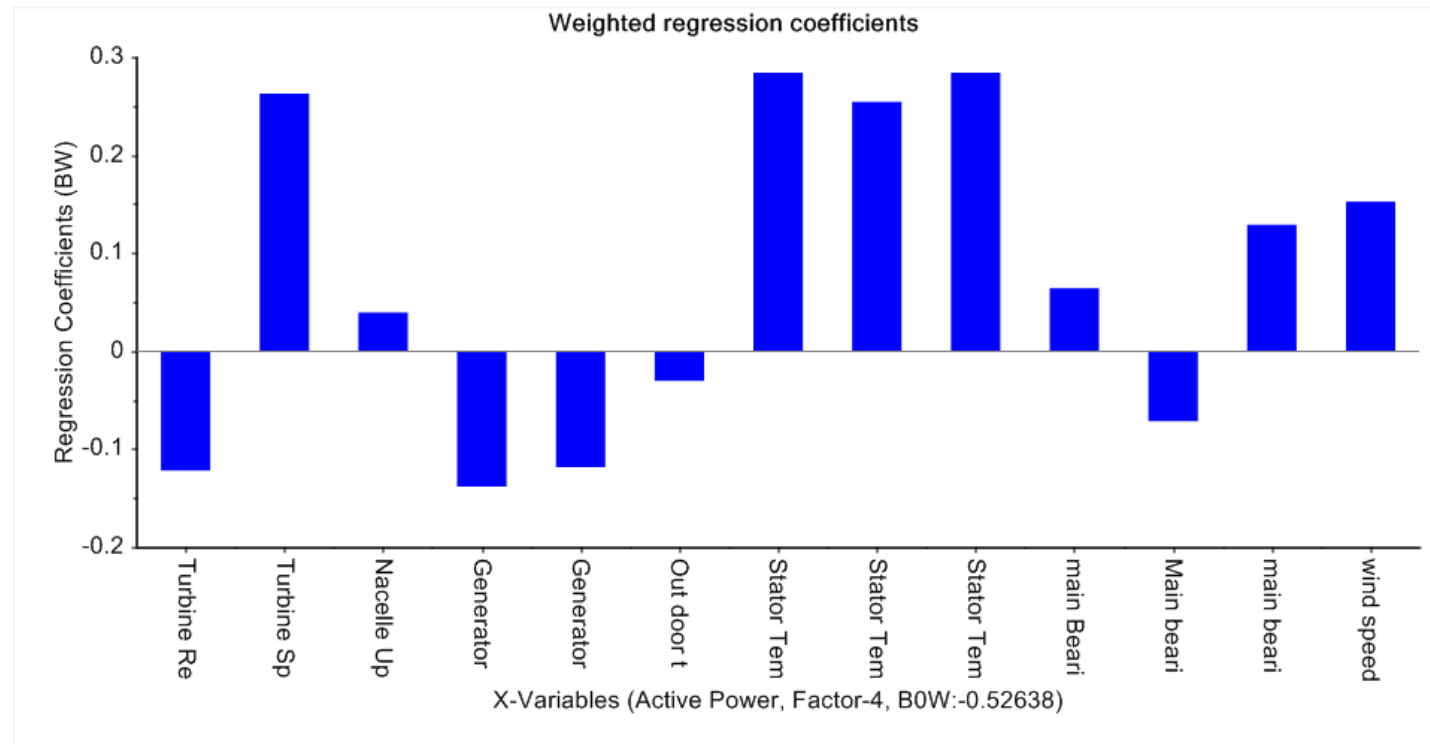
- **RMSEP** =
$$\sqrt{\frac{\sum_{i=1}^N (y - \hat{y}_i)^2}{N}}$$

Root mean squared error of prediction. This is the error to be expected for future predictions, calculated from the validation samples. Expressed in same unit as Y.

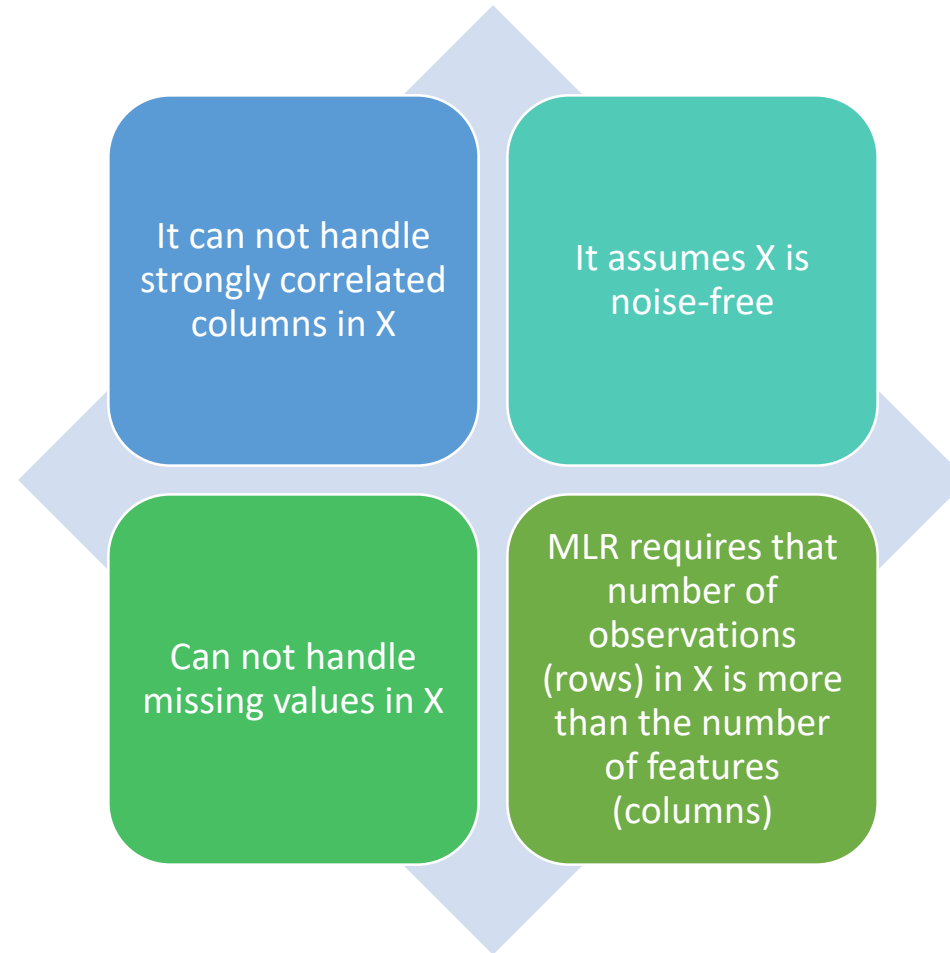
Interpreting Linear Regression Model

Regression coefficients

- Model: $y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_k * x_k + f$
- Each coefficient represents a weighting factor to be applied to the individual variables in order to predict the response.



Shortcomings of MLR

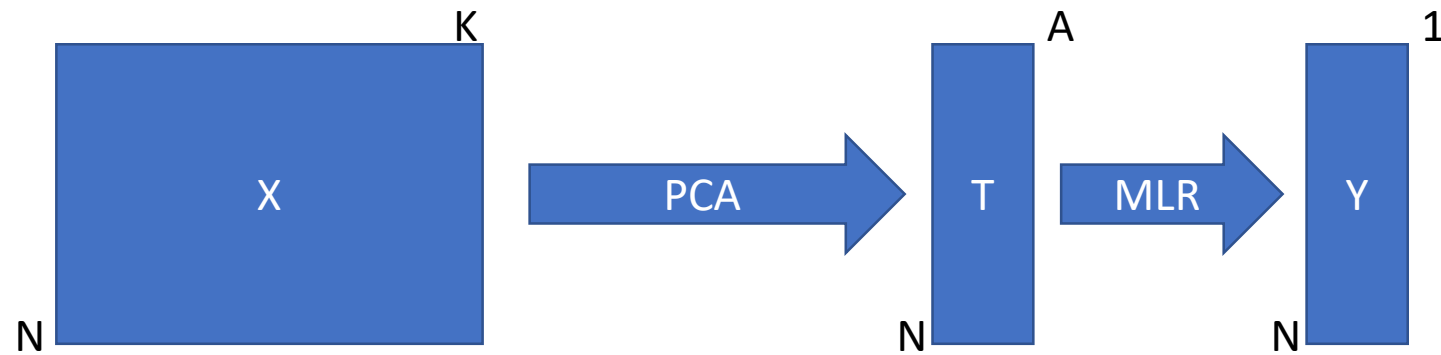


Principal Component Regression

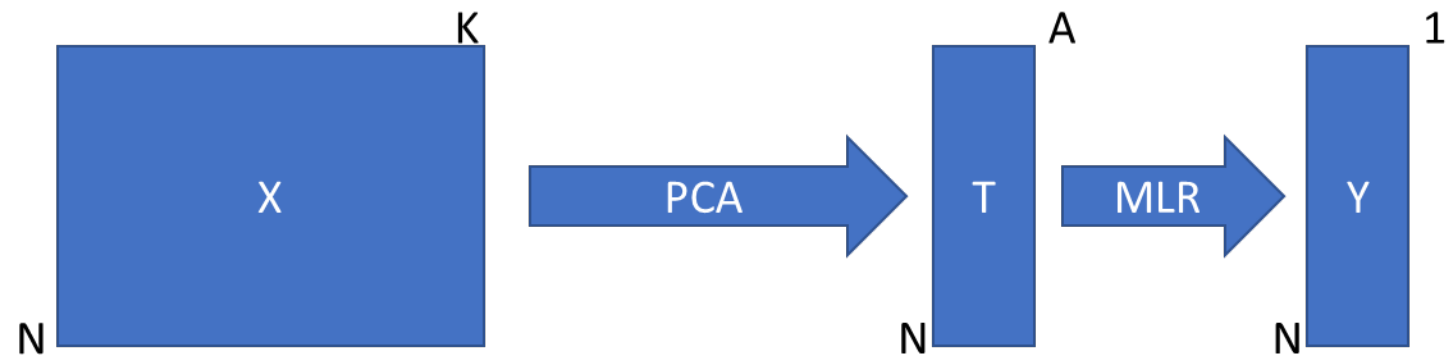
General Idea

Workflow

Octane Example



Advantages of PCR



The columns in T , the scores from PCA, are orthogonal to each other, obtaining independence for the least-squares step.

These T scores can be calculated even if there are missing data in X .

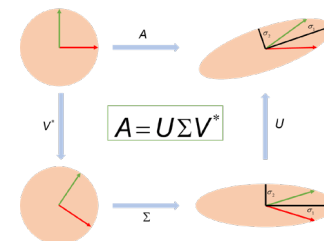
We have reduced the assumption of errors in X , since $X_{\text{reduced}} = TP' + E$. We have replaced it with the assumption that there is no error in T , a more realistic assumption, since PCA separates the noise from the systematic variation in X .

The relationship of each score column in T to vector y can be interpreted independently of each other.

Using MLR requires that $N > K$, but with PCR this changes to $N > A$; an assumption that is more easily met for short and wide X matrices with many correlated columns.

There is much less need to resort to selecting variables from X ; the general approach is to use the entire X matrix to fit the PCA model.

PCA



$$\begin{array}{ccccccc}
 \begin{array}{|c|} \hline X \\ \hline \end{array} & = & \begin{array}{|c|} \hline U \\ \hline \end{array} & \begin{array}{|c|} \hline \Sigma \\ \hline \end{array} & \begin{array}{|c|} \hline V \\ \hline \end{array} & = & \begin{array}{|c|} \hline U \\ \hline \end{array} & \begin{array}{|c|} \hline \Sigma \\ \hline \end{array} & \begin{array}{|c|} \hline V \\ \hline \end{array} & + & \begin{array}{|c|} \hline E \\ \hline \end{array} & = & \begin{array}{|c|} \hline T \\ \hline \end{array} & \begin{array}{|c|} \hline P^T \\ \hline \end{array} & + & \begin{array}{|c|} \hline E \\ \hline \end{array} \\
 m \times n & & m \times m & m \times n & n \times n & & m \times r & r \times r & r \times n & & m \times n & & m \times r & r \times n & & m \times n \\
 & & \text{Rotation} & \text{Stretching} & \text{Rotation} & & & & & & & & & & & \\
 \end{array}$$

Dimensionality reduction

PCR – principal components regression

- basic idea use SVD to form new latent vectors principal components associated with a low rank approximation of X

$$X_{n \times m} = \overbrace{U_{n \times m}^T}^T \overbrace{D_{m \times m} V_{m \times m}'}^{P'} = d_1 u_1 v_1' + d_2 u_2 v_2' + \dots + d_m u_m v_m'$$

- where $U'U = V'V = I$, and D is a diagonal matrix of singular values in descending order of magnitude $d_1 \geq d_2 \geq \dots \geq d_m$
- Columns of T : principal components / factors scores / latent variables
- Columns of V loadings

PCR-Component regression

- Form a low rank approximation of X by keeping just the first $k < m$ principal components (the ones associated with the K largest singular values)

$$X \approx T_k P_k' = d_1 u_1 v_1' + d_2 u_2 v_2' + \dots + d_k u_k v_k'$$

- We now can consider a regression problem in a lower dimensional feature space by using the latent variables as our new features

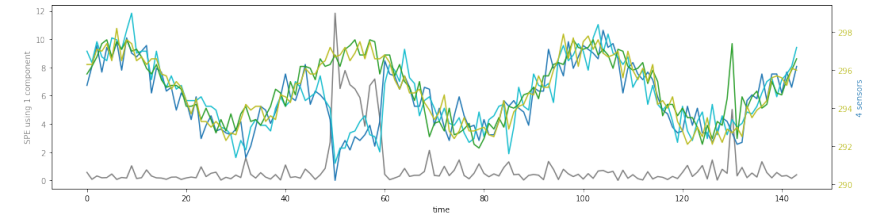
$$Y = T_k C + f$$

$$\hat{C} = (T_k' T_k)^{-1} T_k' Y$$

- Note that the columns of the T are orthogonal to each other (recall $T = UD$) thus $T_k' T_k$ is a diagonal matrix (values on the diagonal are the squares of the singular values) so it is really easy to solve this new regression problem

Steps in building a PCR model

1. Collect the X and y data required for the model.
2. Build a PCA model on the data in X, fitting A components. We usually set A by cross-validation, but often components beyond this will be useful. Iterate back to this point after the initial model to assess if A should be changed.
3. Examine the SPE and T^2 plots from the PCA model to ensure the model is not biased by unusual outliers.
4. Use the columns in T from PCA as your data source for the usual multiple linear regression model (i.e. they are now the X-variables in an MLR model).
5. Solve for the MLR model parameters, $b = (T'T)^{-1}T'y$, an $A \times 1$ vector, with each coefficient entry in b corresponding to each score.



Use the PCR on new observation

1. Obtain your vector of new data, $x'_{new,raw}$ a $A \times 1$ vector.
2. Preprocess this vector in the same way that was done when building the PCA model (usually just mean centering and scaling) to obtain x'_{new}
3. Calculate the scores for this new observation: $t'_{new} = x'_{new} P$
4. Find the predicted value of this observation: $\hat{x}'_{new} = t'_{new} P$
5. Calculate the residual vector: $\hat{e}'_{new} = x'_{new} - \hat{x}'_{new}$
6. Then compute the residual distance from the model plane: $SPE_{new} = \sqrt{\hat{e}'_{new} \hat{e}_{new}}$
7. And the Hotelling's T^2 value for the new observation
8. Before calculating the prediction from the PCR model, first check if the SPE_{new} and T^2_{new} values are below their 95% or 99% limits. If the new observation is below these limits, then go on to calculate the prediction.
9. If either of the SPE or T^2 limits were exceeded, then one should investigate the contributions to SPE , T^2 or the individuals scores to see why the new observation is unusual.

Multiple linear regression, though relatively simpler to implement, has no such consistency check on the new observation's x -values.

Octane example

Unscambler

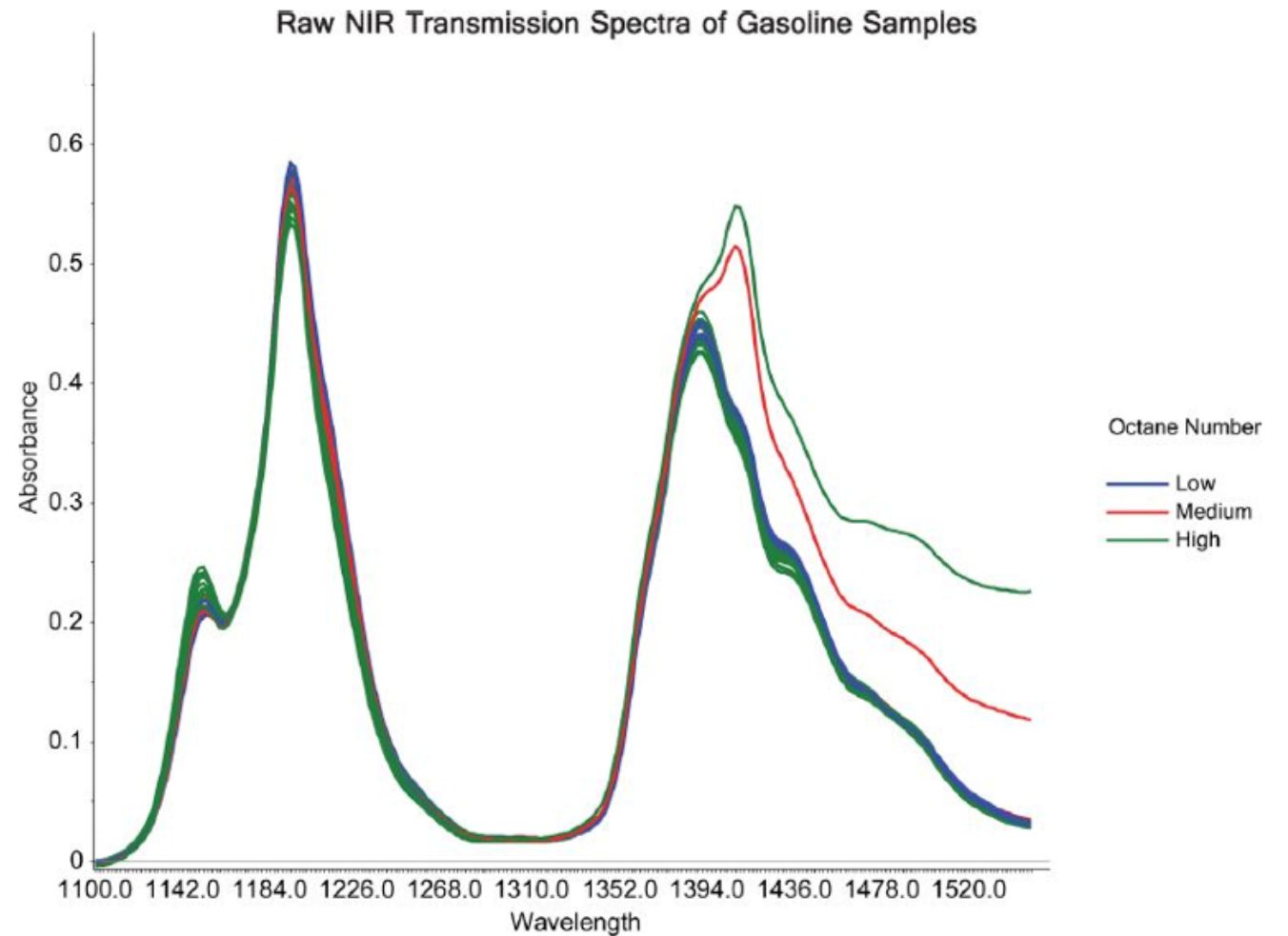
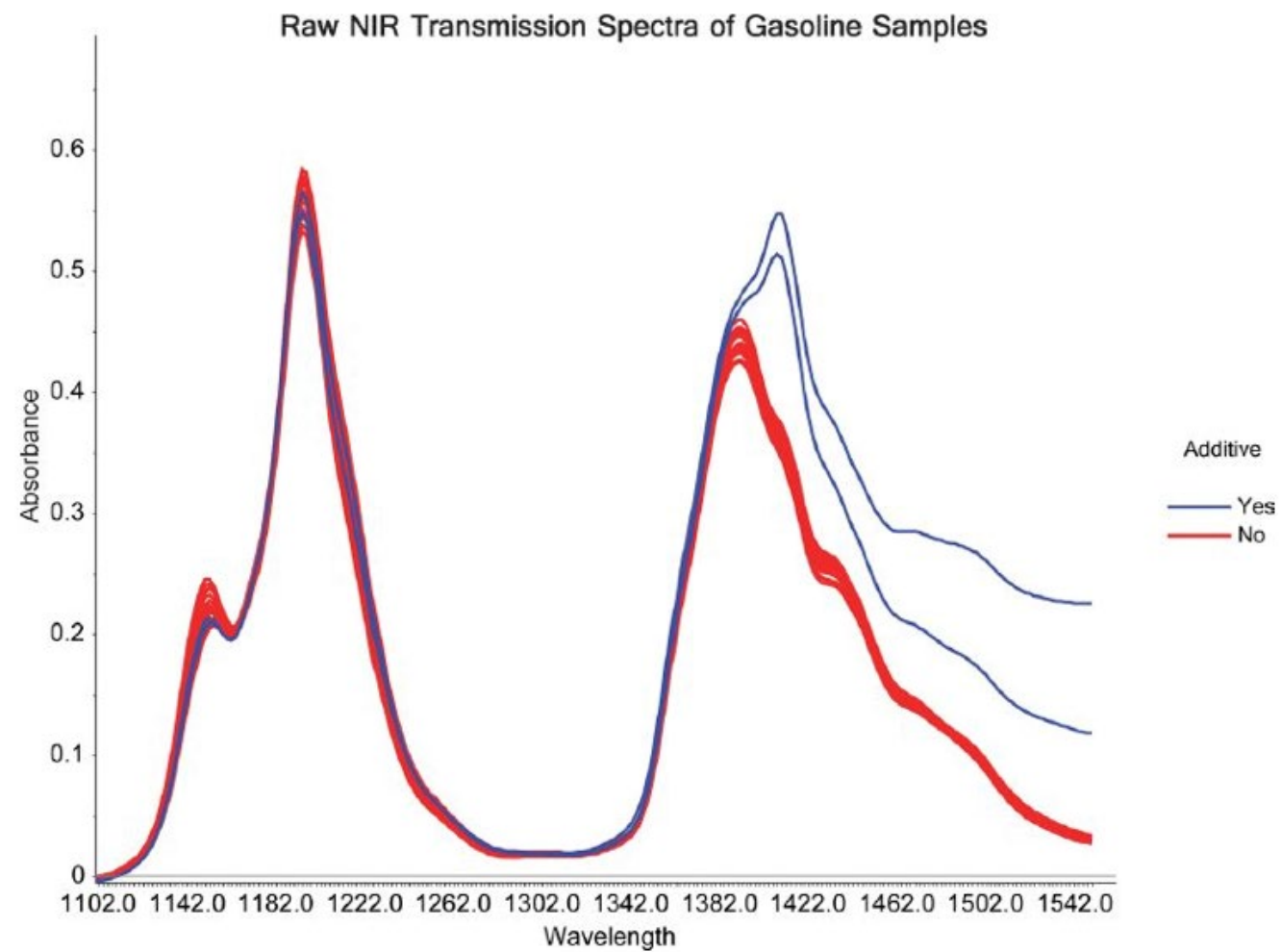
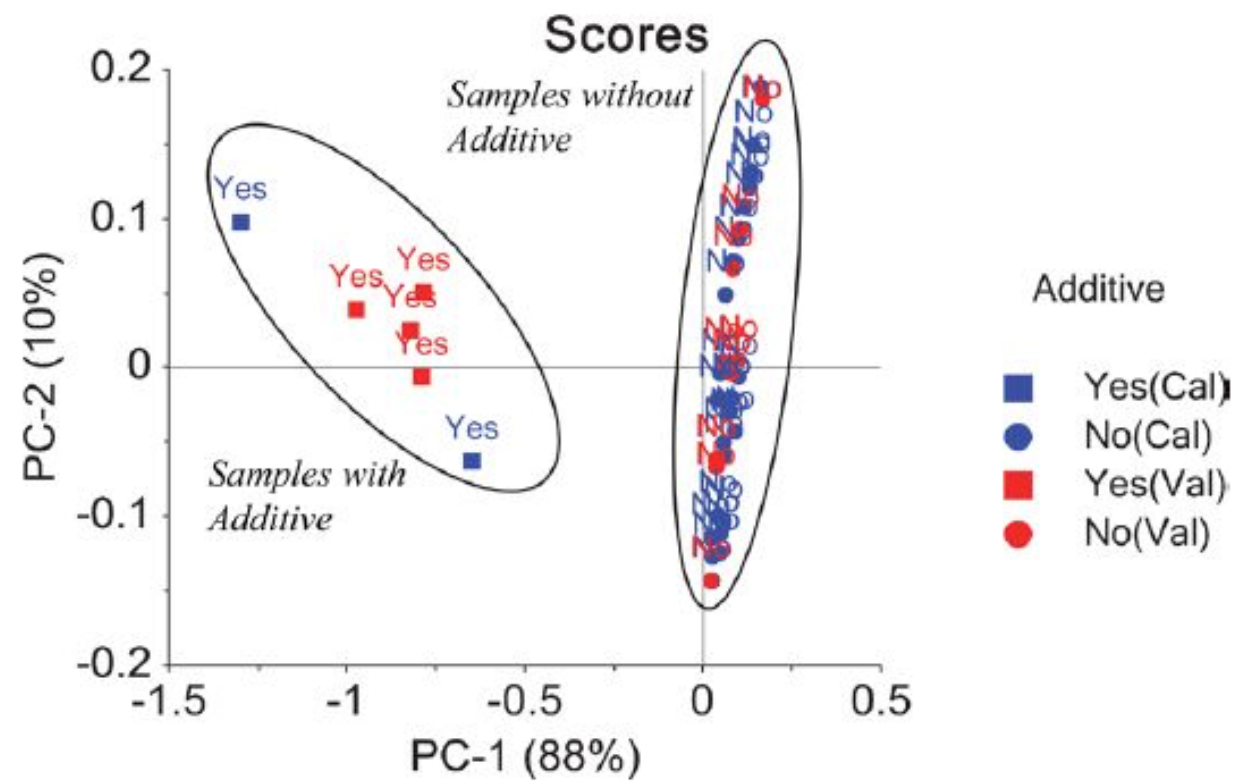


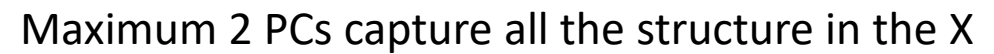
Figure 7.13: Line plot of NIR spectra collected on gasoline samples.

Raw data with label



PCA





PCR vs PLS
NIPALS
Octane Revisited

Partial Least Squares Regression

Issues with PCR

1

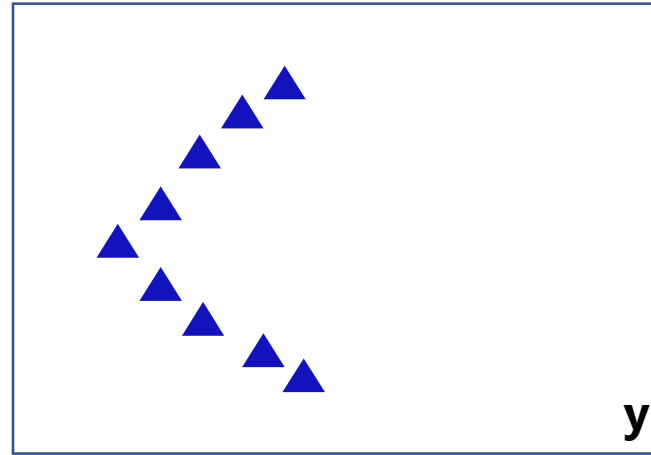
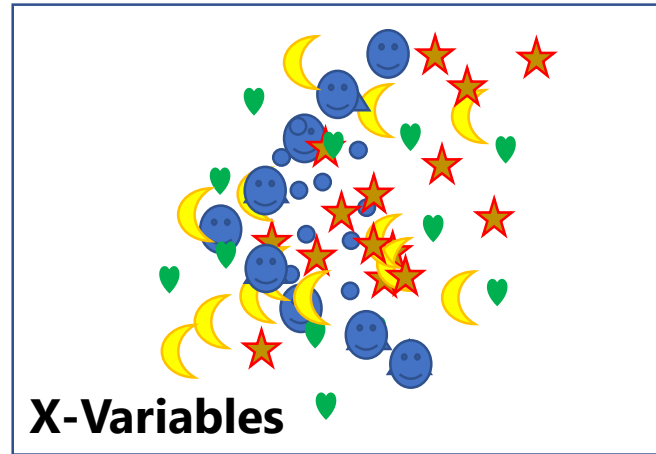
In particular PCR chooses basis vectors of its low dimensional projection to describe as much as possible the variation in the data X

2

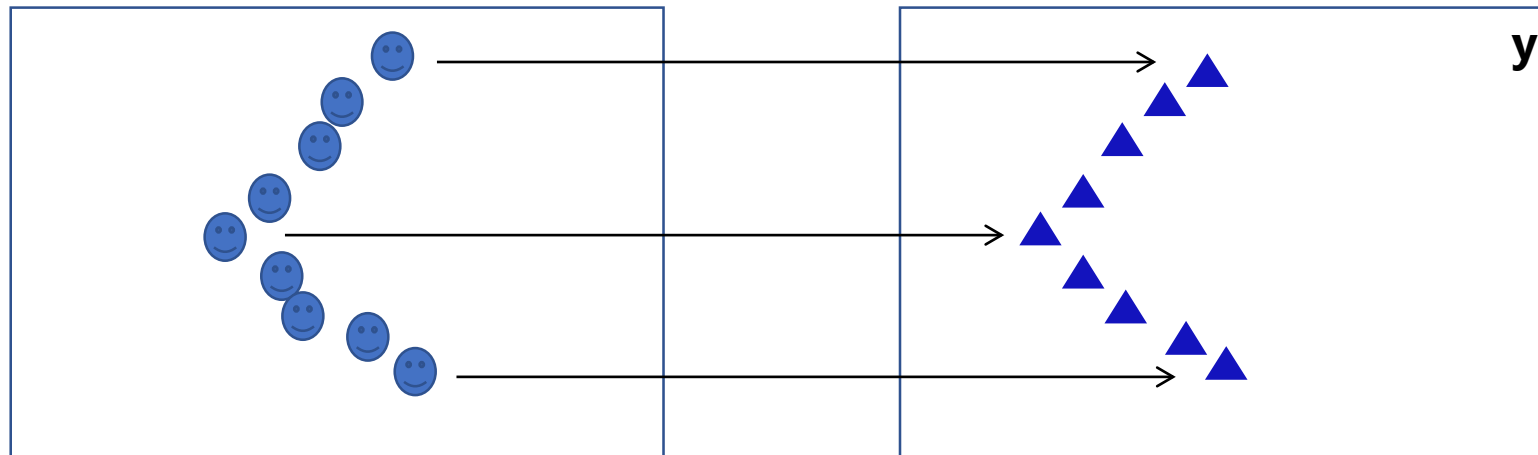
However, nothing guarantees that the principal components which explained X optimally could be relevant for the prediction of Y

Solution: incorporate information from why when choosing the projection full stop with us choose a projection that describes as well as possible the covariation of data X and labels Y

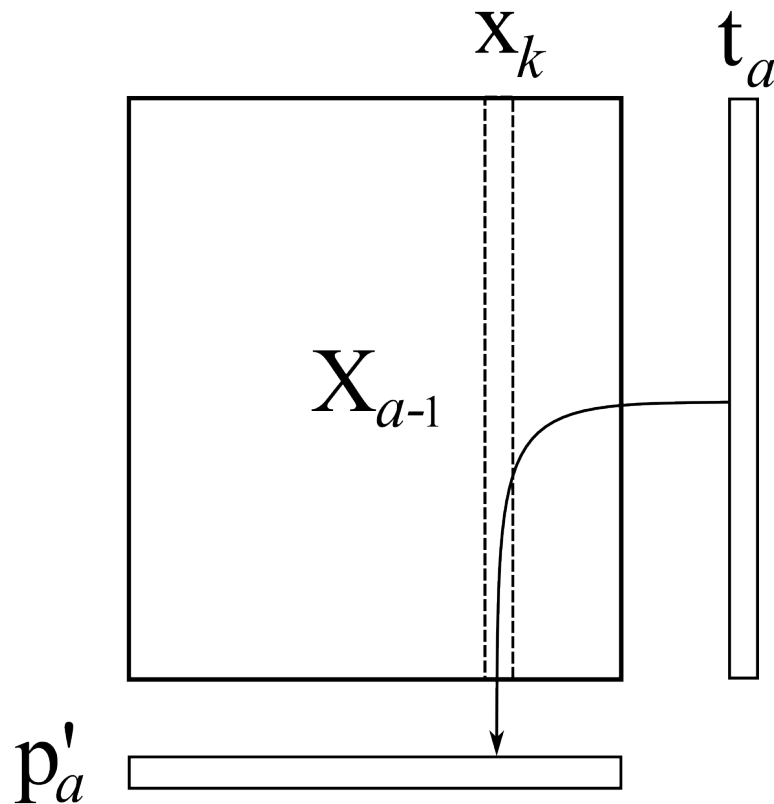
PLS:Extraction of factors: Ideal case



**Extracted Best Correlating
Variable Structure**



Non-linear Iterative Partial Least Squares algorithm



ALGORITHM

Steps to compute PCA using NIPALS algorithm

- Step 1: Initialize an arbitrary column vector \mathbf{t}_a either randomly or by just copying any column of \mathbf{X} .
- Step 2: Take every column of \mathbf{X} , \mathbf{X}_k and regress it onto the \mathbf{t}_a vector and store the regression coefficients as \mathbf{p}_{ka} . (Note: This simply means performing an ordinary least squares regression ($y = mx$) with $x = \mathbf{t}_a$ and $y = \mathbf{X}_k$ with $m = (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y}$). In the current notation we get

$$p_{ka} = \frac{\mathbf{t}'_a \mathbf{X}_k}{\mathbf{t}'_a \mathbf{t}_a}$$

Repeat it for each of the columns of \mathbf{X} to get the entire vector \mathbf{p}_a . This is shown in the illustration above where each column from \mathbf{X} is regressed, one at a time, on \mathbf{t}_a , to calculate the loading entry, p_{ka} . In practice we don't do this one column at a time; we can regress all columns in \mathbf{X} in go:

$$\mathbf{p}'_a = \frac{1}{\mathbf{t}'_a \mathbf{t}_a} \cdot \mathbf{t}'_a \mathbf{X}_a$$

where \mathbf{t}_a is an $N \times 1$ column vector, and \mathbf{X}_a is an $N \times K$ matrix.

- The loading vector \mathbf{p}_a won't have unit length (magnitude) yet. So we simply rescale it to have magnitude of 1.0:

$$\mathbf{p}'_a = \frac{1}{\sqrt{\mathbf{p}'_a \mathbf{p}_a \cdot \mathbf{p}_a}}$$

- Step 4: Regress every row in \mathbf{X} onto this normalized loadings vector. As illustrated below, in our linear regression the rows in \mathbf{X} are our y-variable each time, while the loadings vector is our x-variable. The regression coefficient becomes the score value for that i^{th} row:

$$p_{i,a} = \frac{\mathbf{x}'_i \mathbf{p}_a}{\mathbf{p}'_a \mathbf{p}_a}$$

where \mathbf{x}'_i is a $K \times 1$ column vector. We can combine these N separate least-squares models and calculate them in one go to get the entire vector,

$$\mathbf{t}'_a = \frac{1}{\mathbf{p}'_a \mathbf{p}_a} \mathbf{X} \mathbf{p}'_a$$

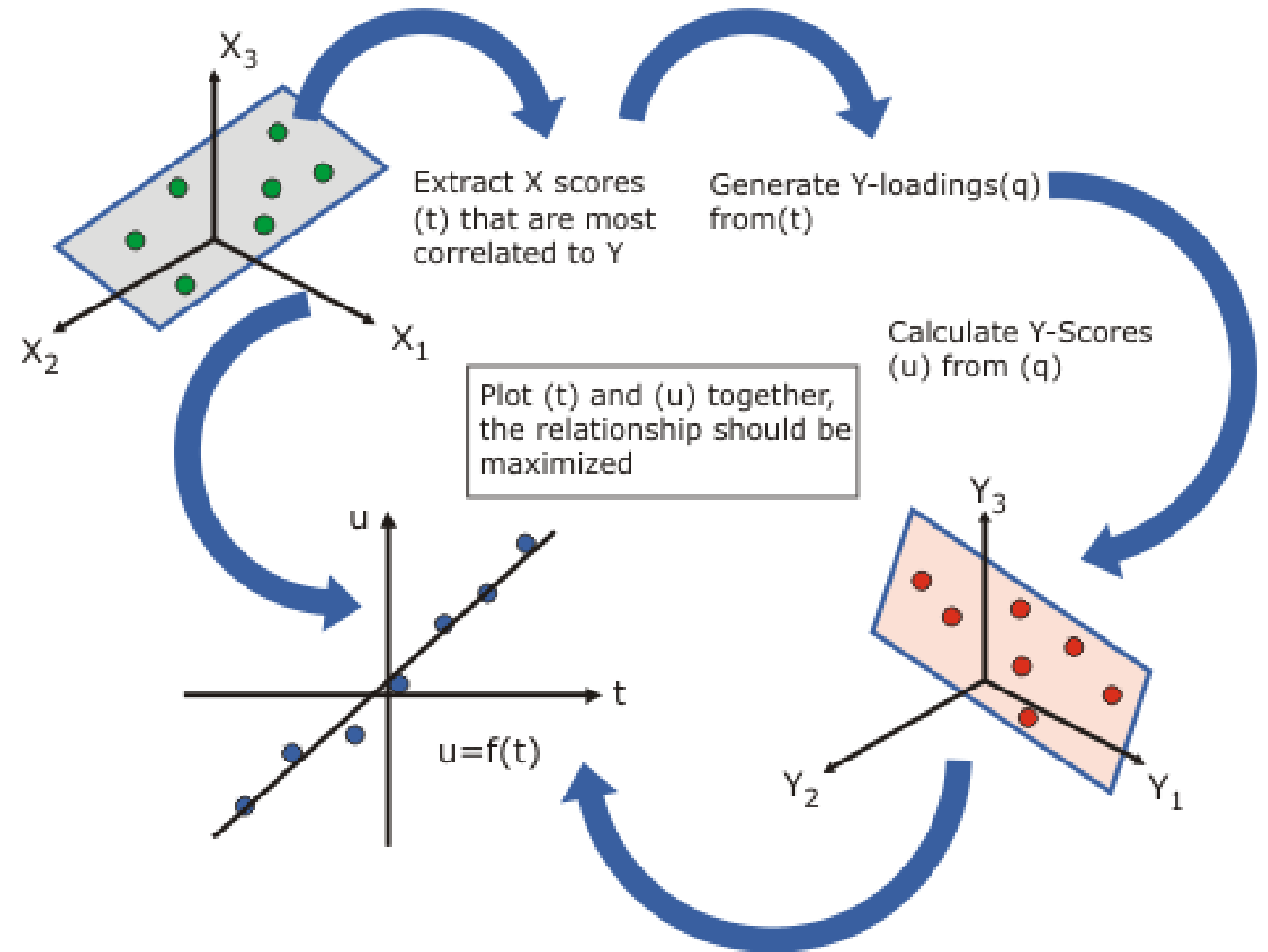
where \mathbf{p}_a is a $K \times 1$ column vector.

- Step 5: Continue looping over steps 2,3,4 until the change in vector \mathbf{t}_a is below a chosen tolerance
- Step 6: On convergence, the score vector and the loading vectors, \mathbf{t}_a and \mathbf{p}_a are stored as the a^{th} column in matrix \mathbf{T} and \mathbf{P} . We then deflate the \mathbf{X} matrix. This crucial step removes the variability captured in this component (\mathbf{t}_a and \mathbf{p}_a) from \mathbf{X} :

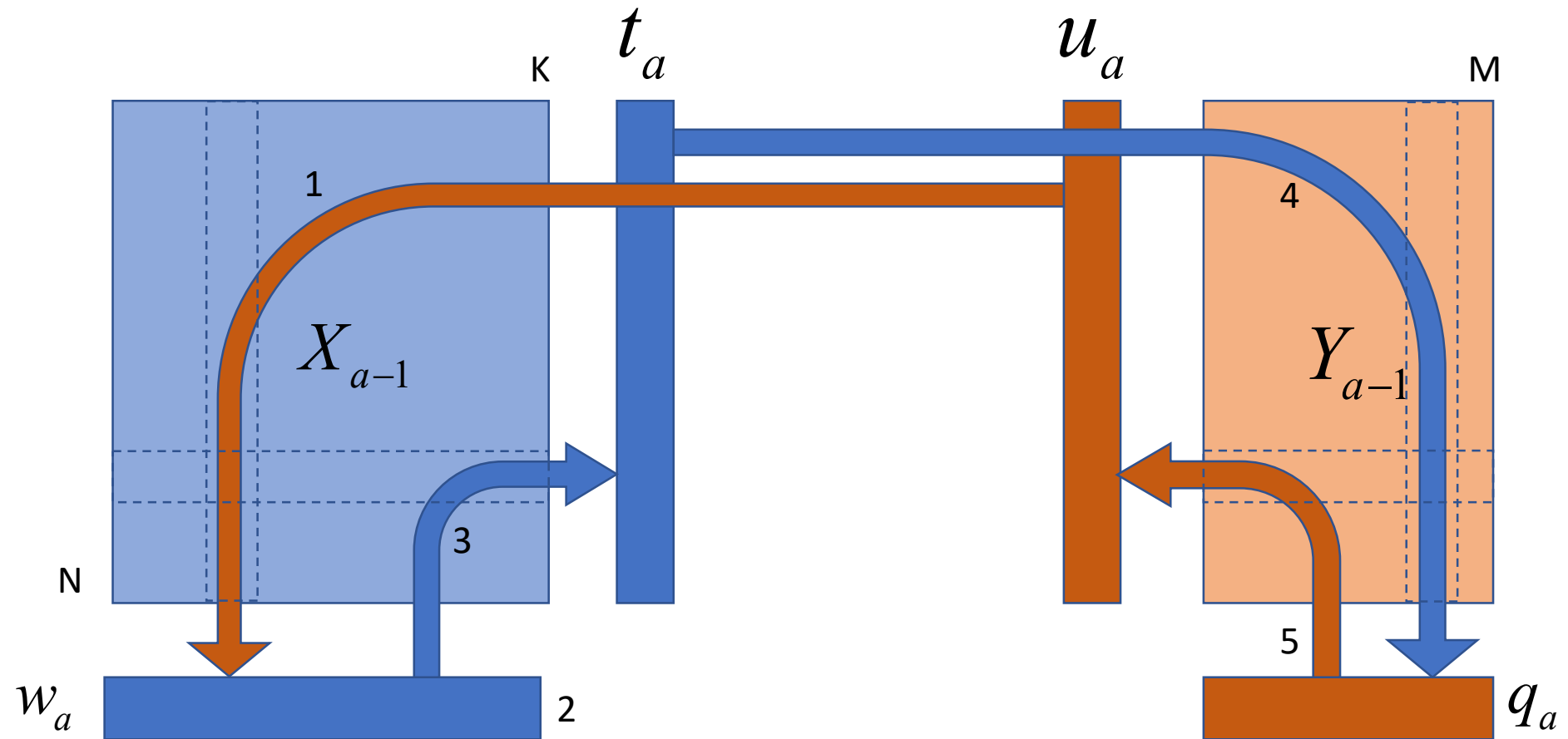
$$\begin{aligned} \mathbf{E}_a &= \mathbf{X}_a - \mathbf{t}_a \mathbf{p}_a' \\ \mathbf{X}_{a+1} &= \mathbf{E}_a \end{aligned}$$

For the first component, \mathbf{X}_a is just the preprocessed raw data. So we can see that the second component is actually calculated on the residuals \mathbf{E}_1 , obtained after extracting the first component. This is called deflation, and nicely shows why each component is orthogonal to the others. Each subsequent component is only seeing variation remaining after removing all the others; there is no possibility that two components can explain the same type of variability. After deflation we go back to step 1 and repeat the entire process for the next component.

Regression modeling Partial Least Squares Squares Regression (PLSR)



NIPALS



NIPALS for PLSR

$$\mathbf{X}_a = \text{normalize}(\mathbf{X})$$

$$\mathbf{Y}_a = \text{normalize}(\mathbf{Y})$$

- Arrow 1: Perform K regressions, regressing each column from \mathbf{X}_a onto the vector \mathbf{u}_a . The slope coefficients from the regressions are stored as the entries in \mathbf{w}_a . Columns in \mathbf{X}_a which are strongly correlated with \mathbf{u}_a will have large weights in \mathbf{w}_a , while unrelated columns will have small, close to zero, weights. We can perform these regression in one go:

$$\mathbf{w}_a = (\mathbf{u}_a' \mathbf{u}_a)^{-1} \mathbf{X}_a' \mathbf{u}_a$$

- Normalize the weight vector to unit length:

$$\mathbf{w}_a = \frac{\mathbf{w}_a}{\sqrt{\mathbf{w}_a' \mathbf{w}_a}}$$

- Regress every row in \mathbf{x}_a onto the weight vector. The slope coefficients are stored as entries in \mathbf{t}_a . This means that rows in \mathbf{x}_a that have a similar pattern to that described by the weight vector will have large values in \mathbf{t}_a . Observations that are totally different to \mathbf{w}_a will have near-zero score values. These N regressions can be performed in one go:

$$\mathbf{t}_a = (\mathbf{w}_a' \mathbf{w}_a)^{-1} \mathbf{X}_a' \mathbf{w}_a$$

- Arrow 4 Next, regress every column in \mathbf{y}_a onto this score vector, \mathbf{t}_a . The slope coefficients are stored in \mathbf{q}_a . We can calculate all M slope coefficients:

$$\mathbf{q}_a = (\mathbf{t}_a' \mathbf{t}_a)^{-1} \mathbf{Y}_a' \mathbf{t}_a$$

Arrow 5 Finally, regress each of the N rows in \mathbf{y}_a onto this weight vector, \mathbf{q}_a . Observations in \mathbf{y}_a that are strongly related to \mathbf{q}_a will have large positive or negative slope coefficients in vector \mathbf{u}_a :

$$\mathbf{u}_a = (\mathbf{q}_a' \mathbf{q}_a)^{-1} \mathbf{Y}_a' \mathbf{q}_a$$

This is one round of the NIPALS algorithm. We iterate through these 4 arrow steps until the \mathbf{u}_a vector does not change much. On convergence, we store these 4 vectors: \mathbf{w}_a , \mathbf{t}_a , \mathbf{q}_a and \mathbf{u}_a which jointly define the a^{th} component. Then we deflate. Deflation removes variability already explained from \mathbf{X}_a and \mathbf{Y}_a . Deflation proceeds as follows:

Step 1: Calculate a loadings vector for the X space, \mathbf{p}_a called using the X -space scores:

$$\mathbf{p}_a = (\mathbf{t}_a' \mathbf{t}_a)^{-1} \mathbf{X}_a' \mathbf{t}_a$$

Step 2: Remove the predicted variability from X and Y Using the loadings, \mathbf{p}_a just calculated above, we remove from \mathbf{X}_a the best prediction of \mathbf{p}_a , in other words, remove everything we can explain about it. We also remove any variance explained from \mathbf{Y}_a :

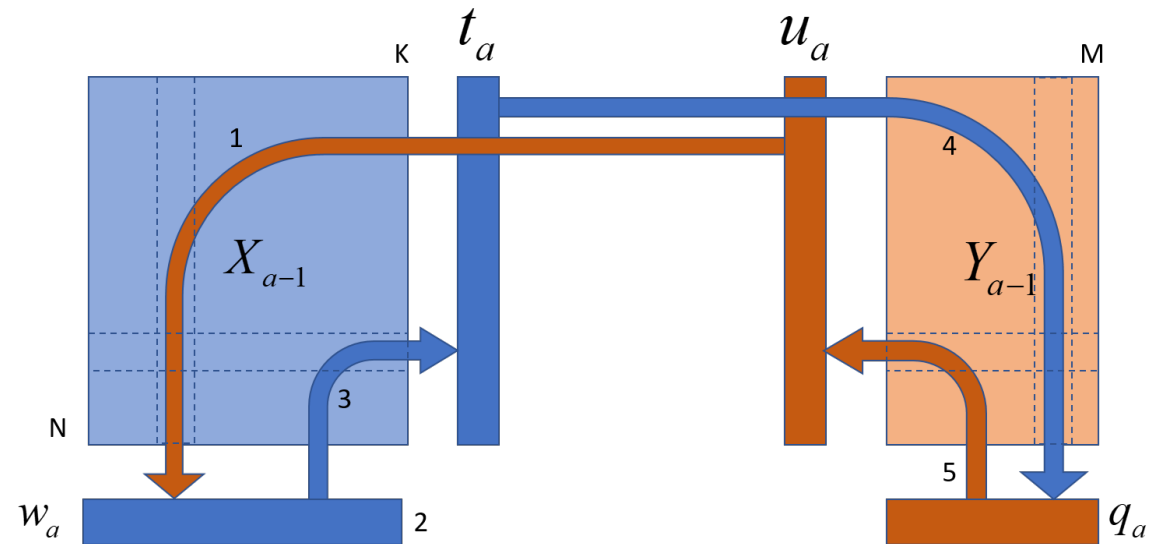
$$\hat{\mathbf{X}}_a = \mathbf{t}_a \mathbf{p}_a'$$

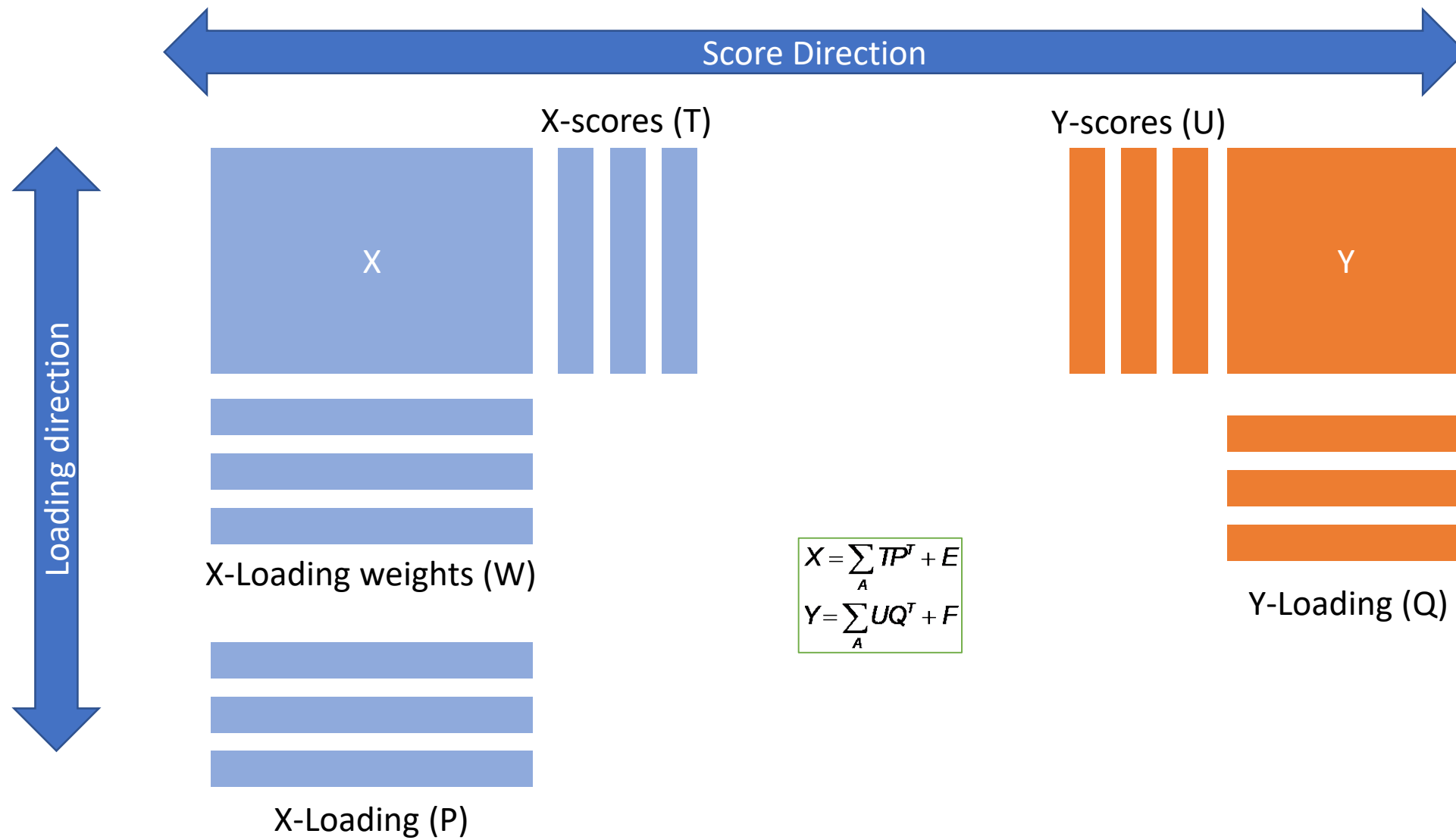
$$\hat{\mathbf{Y}}_a = \mathbf{t}_a \mathbf{q}_a'$$

$$\mathbf{X}_a = \mathbf{X}_a - \hat{\mathbf{X}}_a$$

$$\mathbf{Y}_a = \mathbf{Y}_a - \hat{\mathbf{Y}}_a$$

[Python notebook](#)





PLSR terminology and model structure

Scores: (X-scores: T , Y-scores: U) Map of samples. Locations of objects when projected onto the factors

Loadings: (X-loadings: P , Y-loadings: Q) Map of variables. Describe the structure of X or Y

Loading weights: (X-loading weights: W) Describe relationships between X- and Y-variables with the purpose of prediction

Residuals: (X-residuals: E , y-residuals: F) Error

Variance: Mean squares of residuals = residual variance

Model equations: $X = TP^T + E$ and $Y = TQ^T + F$

Regression coefficients: $Y = B_0 + X_1 * B_1 + X_2 * B_2 + \dots + X_N * B_N$

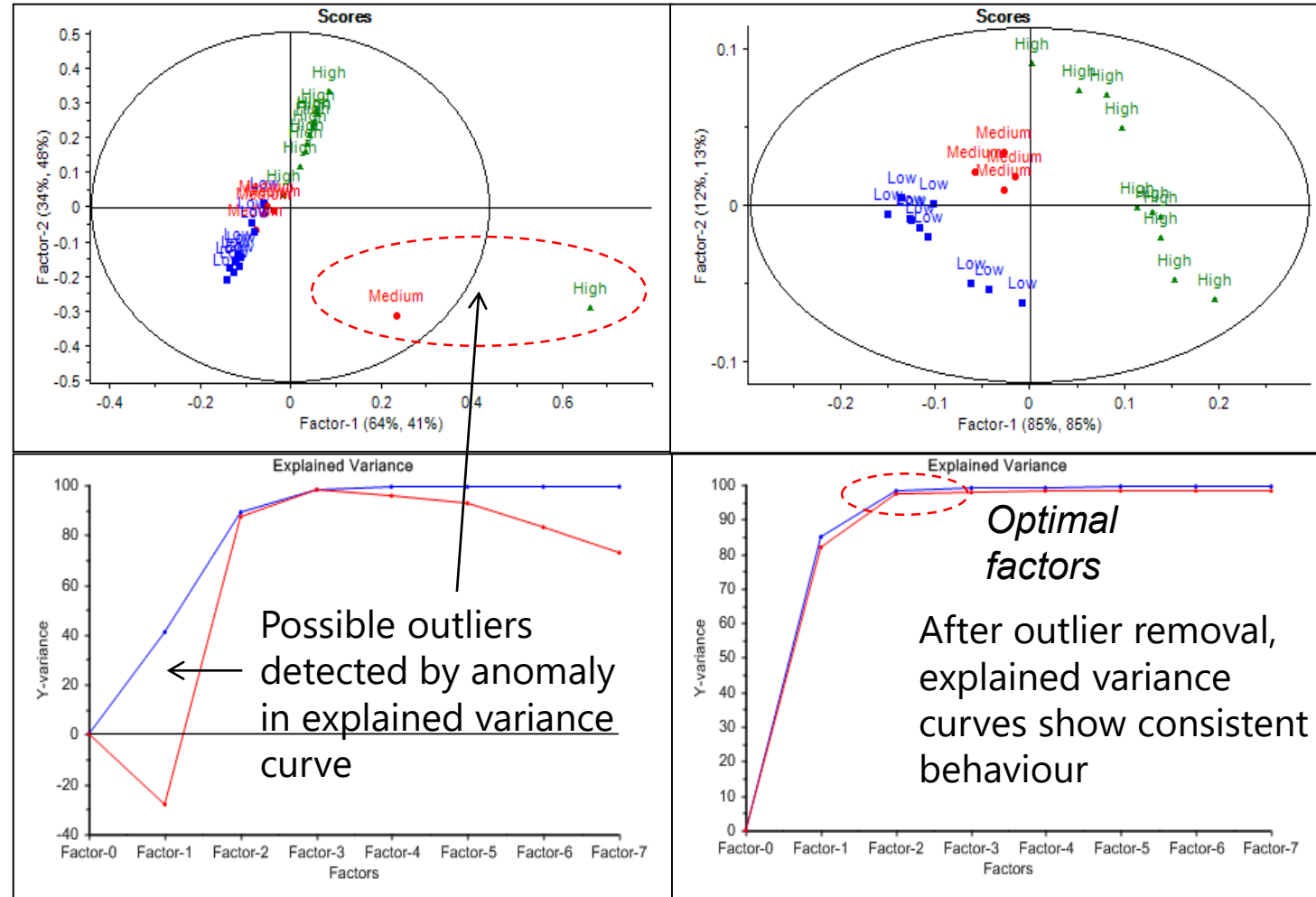
Where B is given by $B = W(P^TW)^{-1}Q^T$

Regression modeling

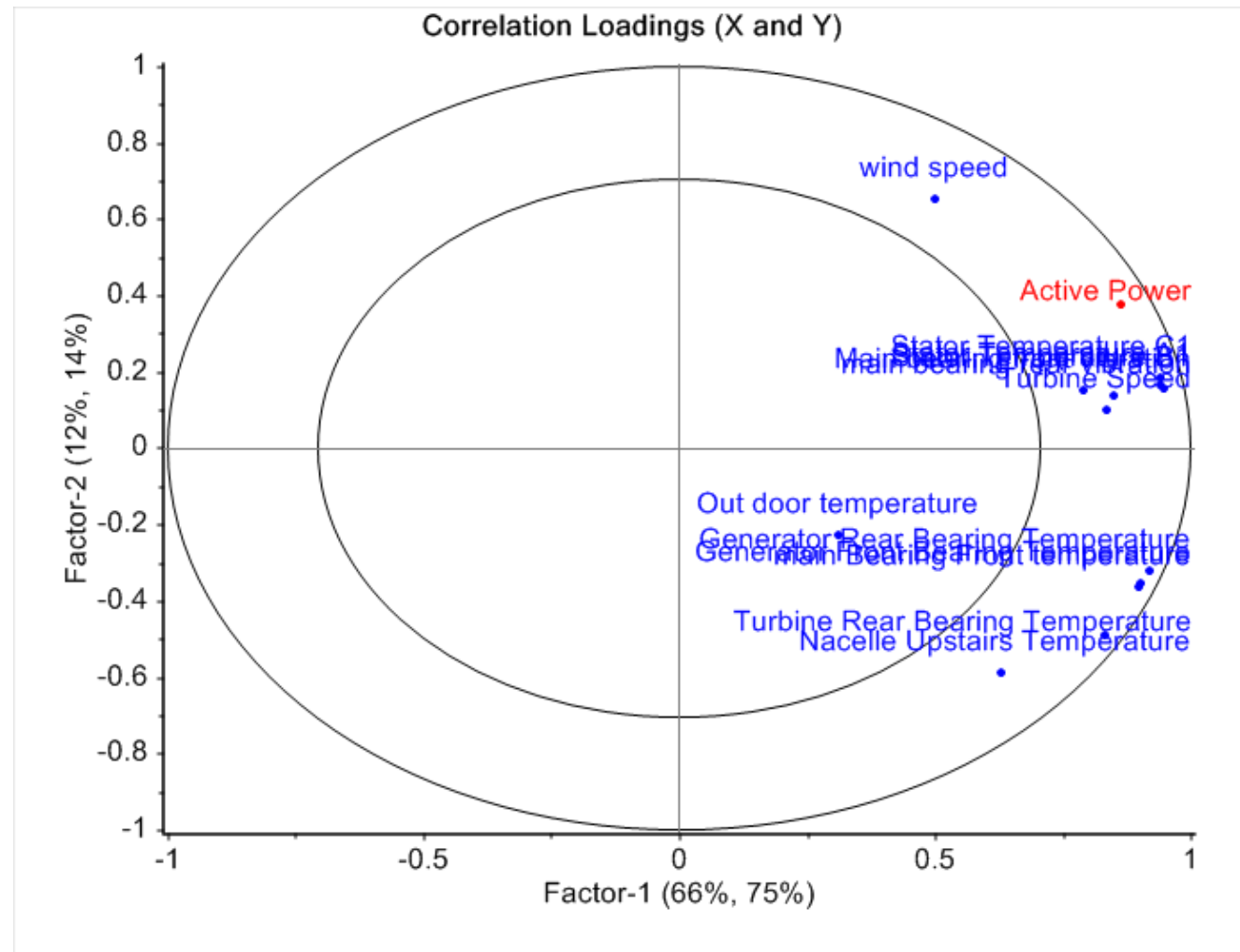
Advantages of latent variable regression

- PCR and PLSR are most useful (compared to MLR) when
 - The number of X-variables is large (i.e. $\gg 30$)
 - There are more than one Y-variable
 - The X-variables are correlated
- PCR and PLSR offer the following advantages over MLR
 - They provide the same diagnostic plots as are found in PCA
 - It is much easier to interpret sample and variable relationships
- PLSR ensures more relevant latent variables
 - Simpler models
 - Easier interpretation

Interpreting Regression Models: Scores

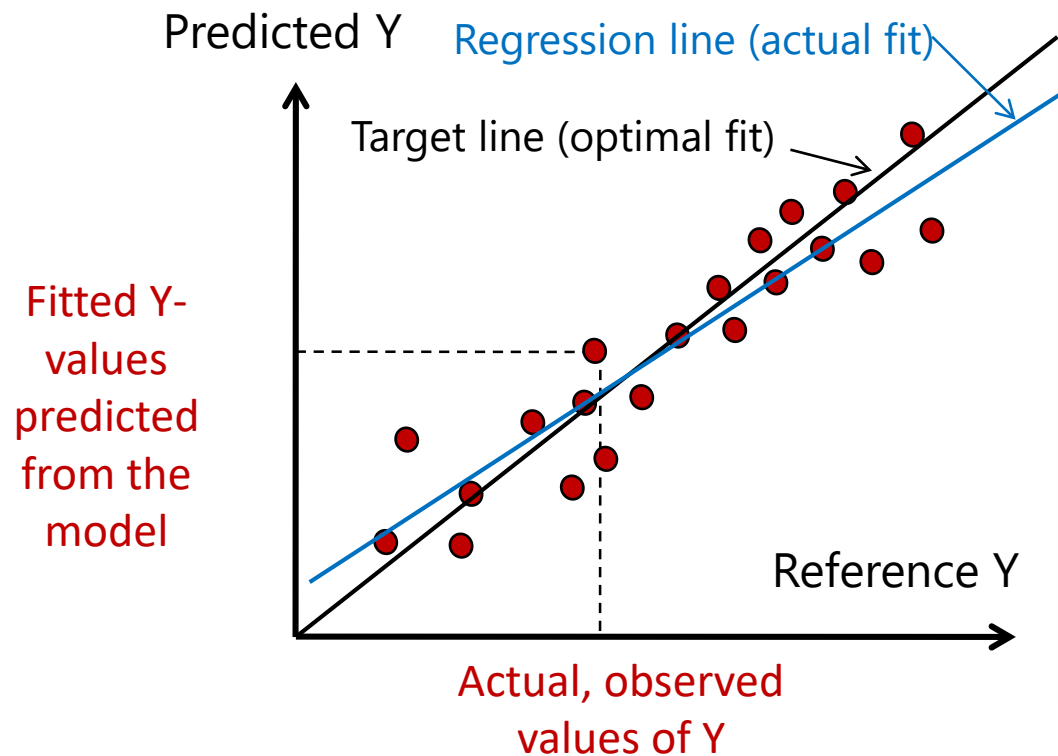


Interpreting Regression Models: Correlation loading



Interpreting Regression Models : Predicted vs. Reference plot

The predicted vs. reference plot shows how close the fitted response values are to the observed ones.



Summary statistics:

Slope: Slope of regression line

Offset: Intercept of the regression line

Correlation: Taken between Y and \hat{Y}

R2(Pearson): Squared Correlation

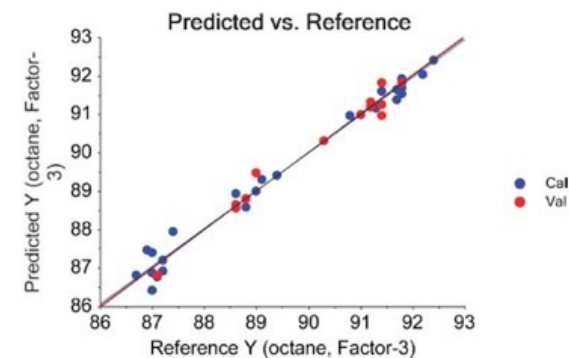
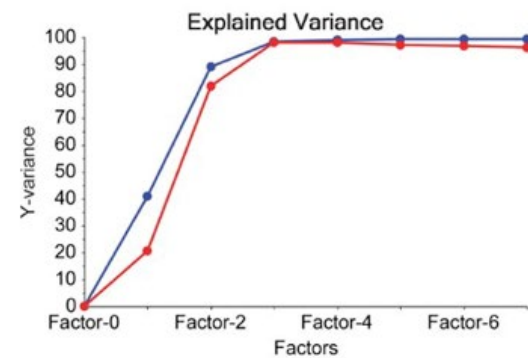
R-square: Explained variance

RMSEC(/-E/-CV/-P) : Calibration (validation) error, expressed in units of original response values

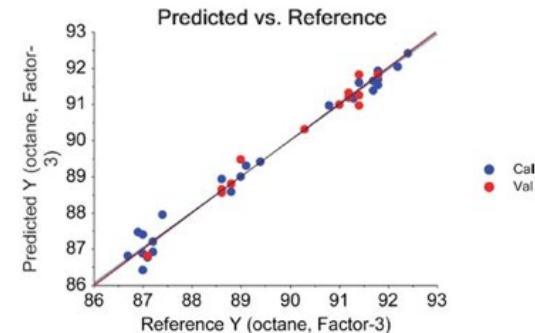
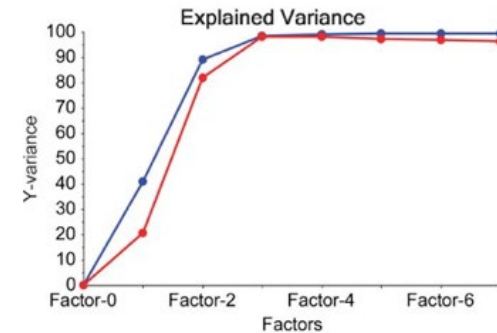
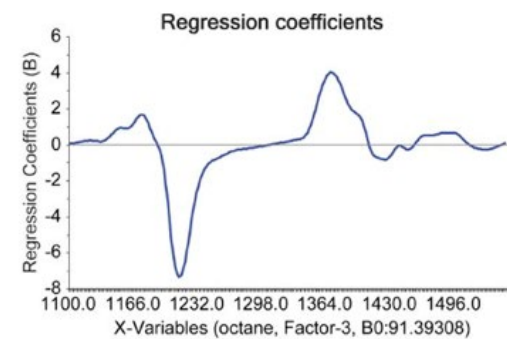
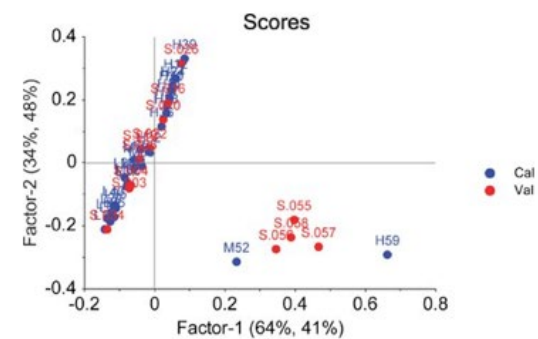
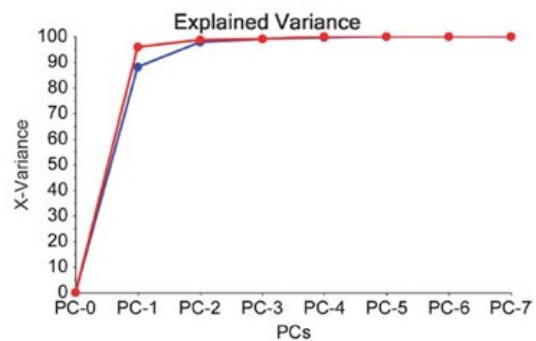
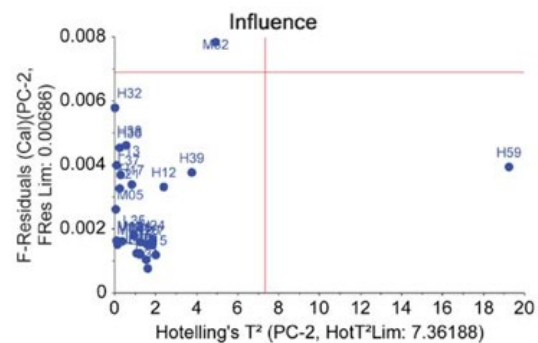
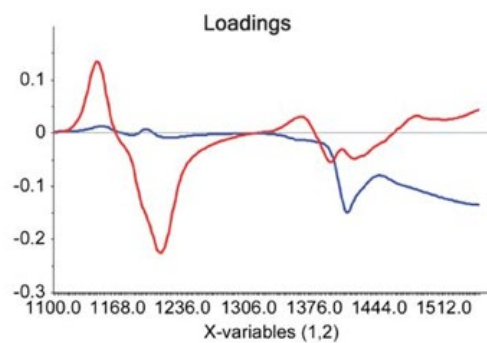
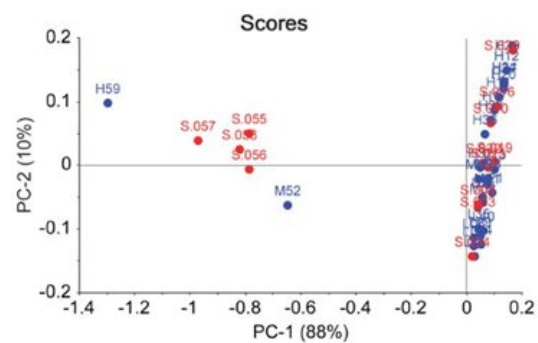
SEC(/-E/-CV/-P): Standard error of calibration (validation), found from the SDev of residuals

Bias: Average value of residuals

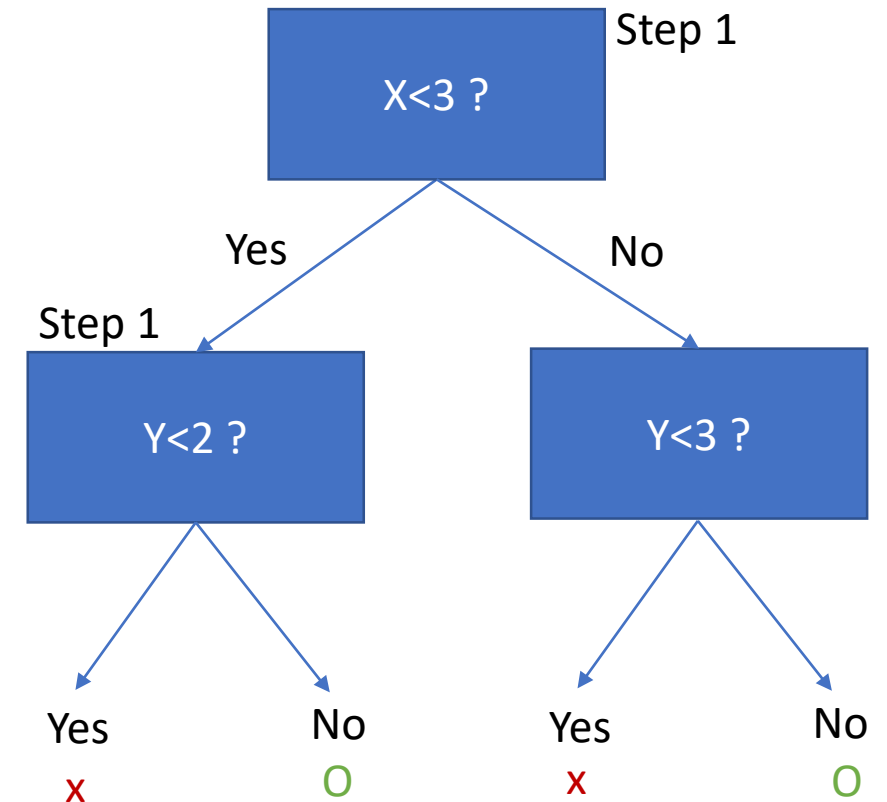
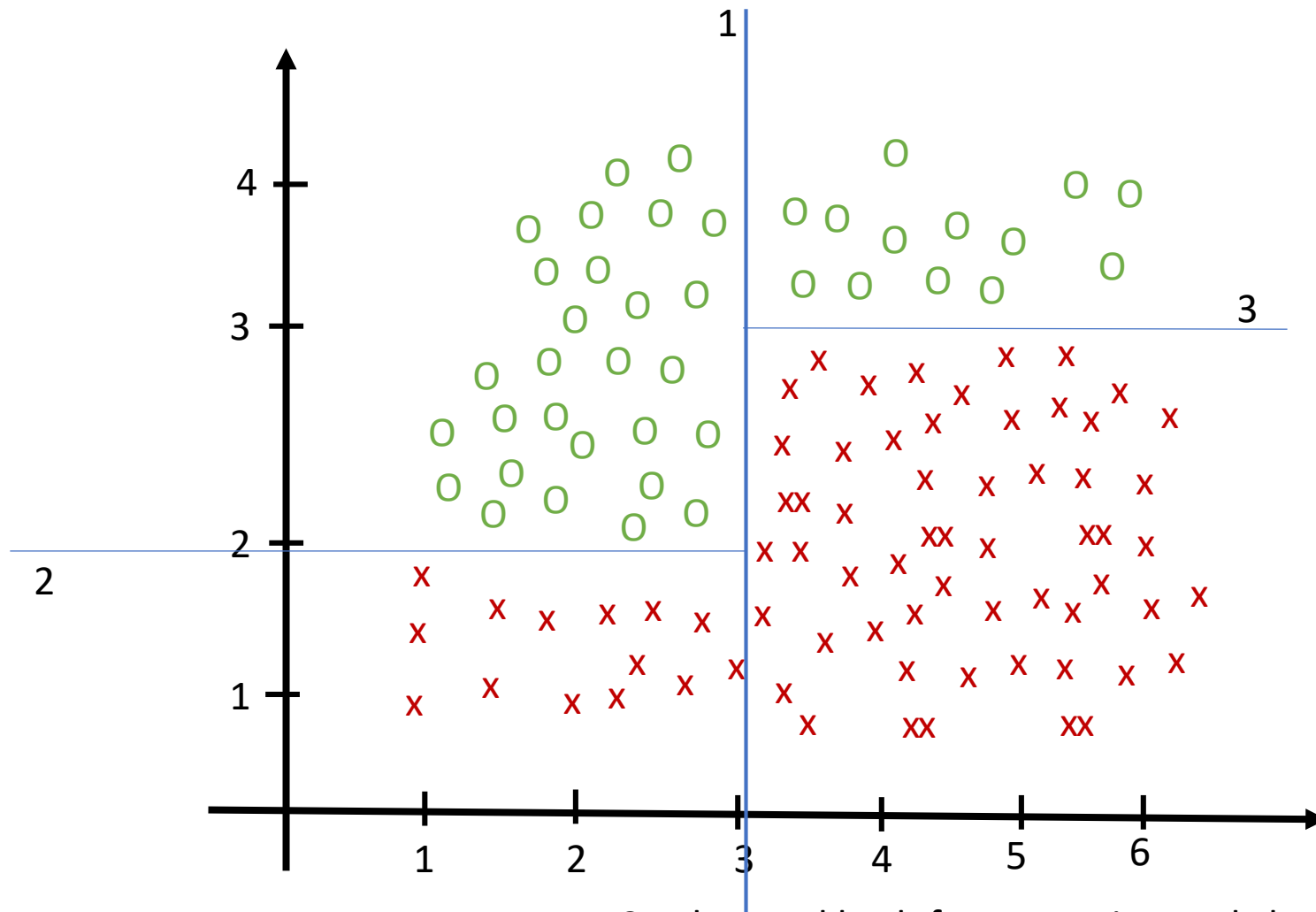
Octane data revisited



PCR vs PLSR



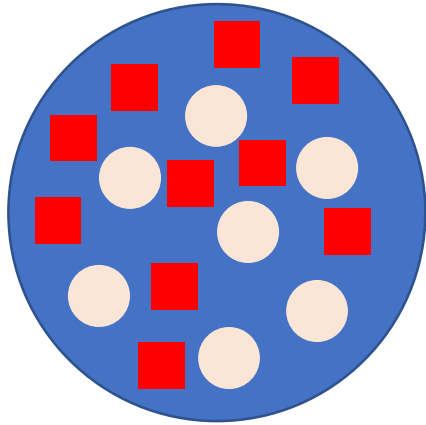
Bonus: Decision Trees and Random Forest



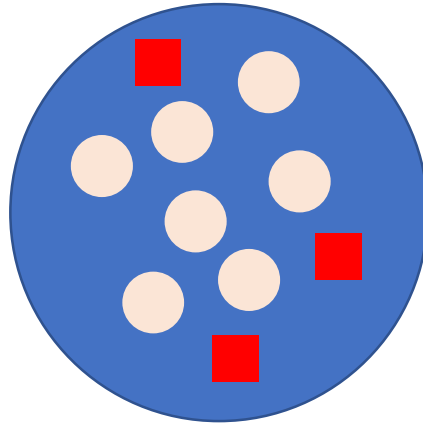
Asks Multiple Linear Question until the data is fully classified

Can be used both for regression and classification
But on what basis should I split the data ?

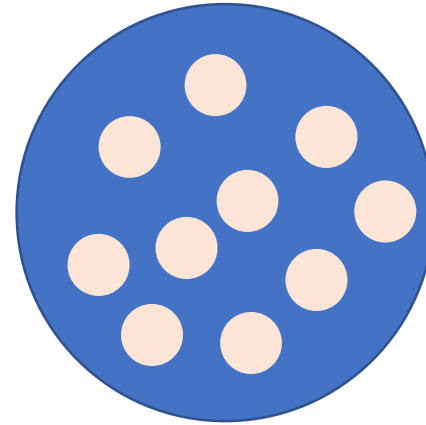
Impurity, Entropy and Information Gain



Very Impure



Impure



Pure

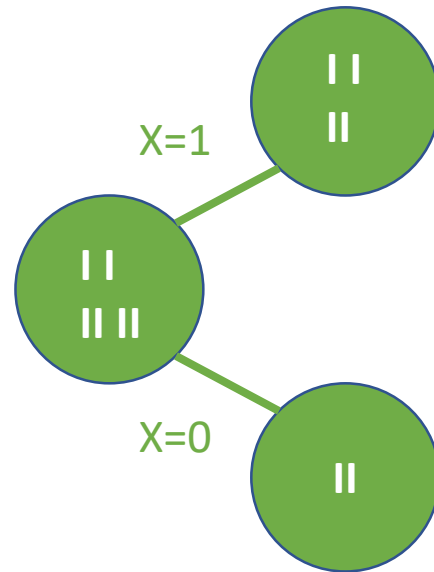
$$Entropy = \sum_i -p_i \log_2 p_i$$

p_i is the probability of class i

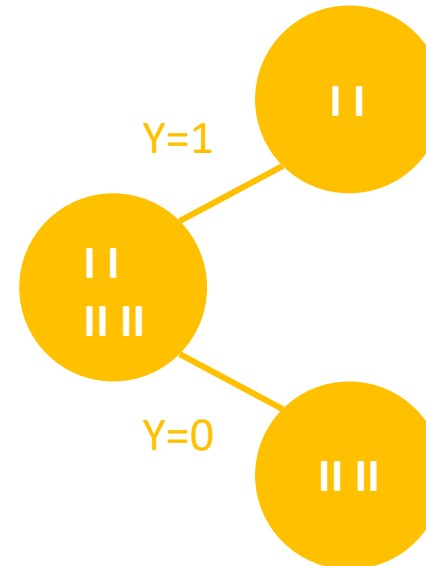
$$Information\ Gain = Entropy_{parent} - Average\ Entropy_{children}$$

Using IG to construct Decision Trees

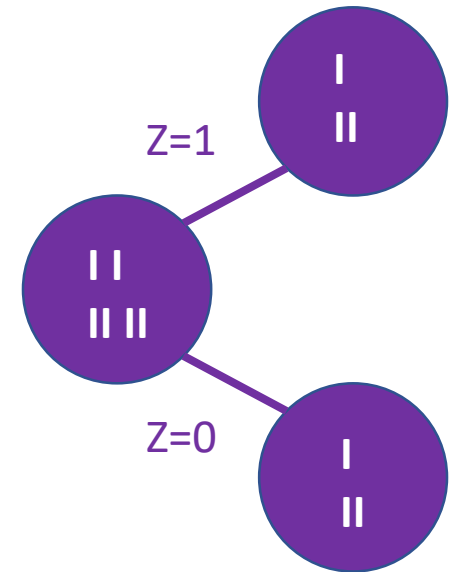
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II



IG=0.3112



IG=1



IG=0

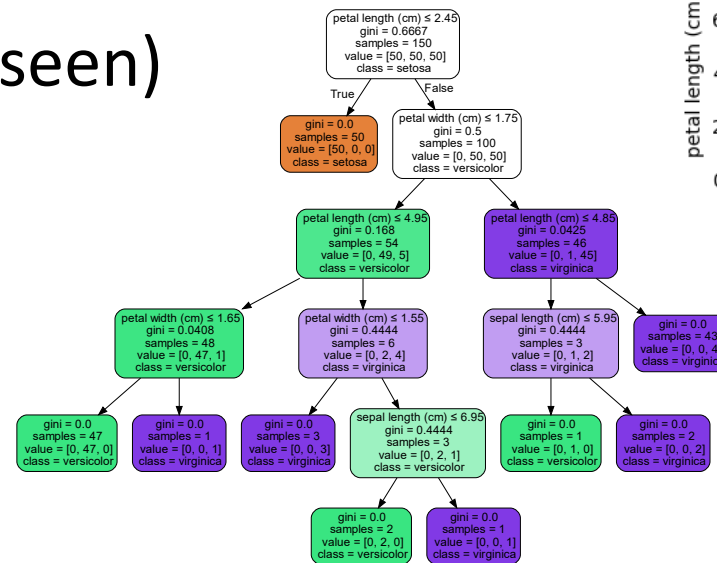
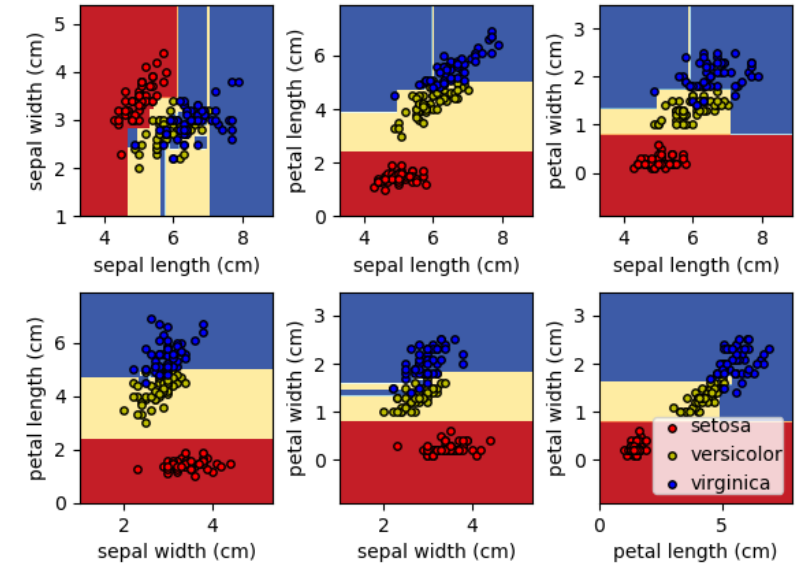
Split on Y attribute

Choose the attribute with highest information gain for the full training set at the root of the tree.

Decision Trees: Code

```
from sklearn import tree  
clf = tree.DecisionTreeClassifier()  
clf = clf.fit(X, Y)  
Yunseen=clf.predict(Xunseen)
```

Decision surface of a decision tree using paired features



Decision Trees: + vs -

- Advantages
 - Interpretable
 - Requires little data preprocessing
 - The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
 - Automatic feature detection
- Disadvantages
 - Overfitting due to complex tree formation.
 - Small variations in the data might result in a completely different tree being generated.
 - Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Excercises

- Unscambler download data from blackboard
 - Weight, height, shoesize
 - Octane data
- [Python notebook](#)
 - Start from the Linear Regression notebook and implement PCA, PCR and PLS
 - You can load the data in Unscambler too.
 - Met training and test data can be found on blackboard
- Read about SVM and Decision Tree
- Prepare for the flip classroom next week