McMaster University ENGINEERING

Chemical Engineering 4H03

Projection of Latent Structures (NIPALS)

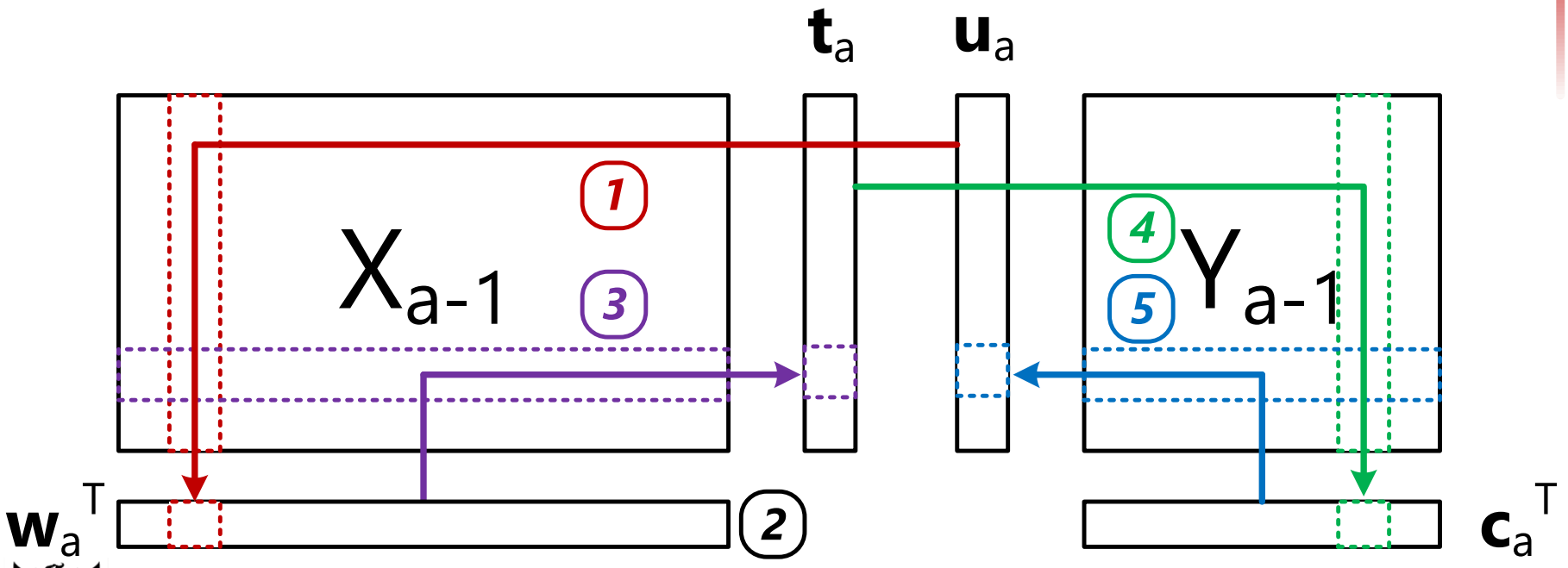
Jake Nease
McMaster University

Where are We?

- We have derived PLS from “first principles”
  - Well, not really, but we know the **objective**
- Unfortunately, without optimization we have no way of reliably computing $\boldsymbol{c}_a$ and $\boldsymbol{w}_a$ to $\max \psi = \mathcal{C}(\boldsymbol{t}_a, \boldsymbol{u}_a)$
  - I’ll gently remind you here that $\max \psi$ is equivalent to finding the “pareto-optimal” of our three competing objectives
- This leads us to adapting our NIPALS algorithm for PLS

NIPALS: The Idea

- Imagine creating PCA models for the X and Y spaces separately using NIPALS
  - What would that look like?
  - What is the main thing missing?
- PLS NIPALS looks very similar to PCA:



McMaster University ENGINEERING

NIPALS Algorithm for PLS

Naturally, **I**'d **P**refer **A**nother **L**east-**S**quares

PLS NIPALS Algorithm

- NIPALS begins with X_0 as before AND Y_0
  - Recall X_a and Y_a are the data sets explained by a components

FOR $a = 1, 2, \cdots, A$:

1. Select an (arbitrary) column as $\boldsymbol{u}_a$
2. In a loop until convergence:
   - I. Regress columns from X_{a-1} onto $\boldsymbol{u}_a$ to get weights $\boldsymbol{w}_a$
   - II. Normalize $\boldsymbol{w}_a$ to unit length
   - III. Regress rows from X_{a-1} onto $\boldsymbol{w}_a$ to get $\boldsymbol{t}_a$
   - IV. Regress columns from Y_{a-1} onto $\boldsymbol{t}_a$ to get loadings $\boldsymbol{c}_a$
   - V. Regress rows from Y_{a-1} onto $\boldsymbol{c}_a$ to get $\boldsymbol{u}_a$
3. Deflate component from X_{a-1} and Y_{a-1} to get X_a and Y_a

END

NIPALS for PLS Step By Step

- STEP 1 Select an arbitrary column for $\boldsymbol{u}_a$
  - Any individual column in Y_0
  - A column of normally distributed random numbers
  - Basically anything except the zero column (WHY??)

NIPALS for PLS Step By Step

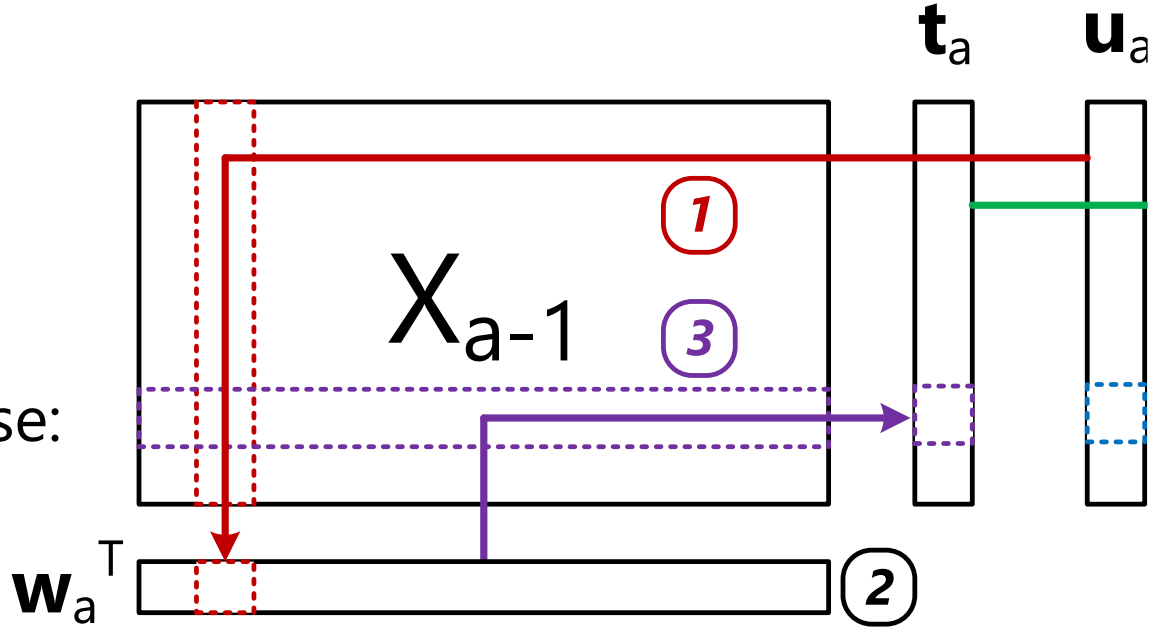
- STEP 2.1 Regress each column of X_{a-1} ($\boldsymbol{x}_k$) onto $\boldsymbol{u}_a$
  - Perform a LS regression of $\boldsymbol{x}_k$ onto $\boldsymbol{u}_a$ *(regress y onto x)*
  - Store the regression coefficient as $w_{k,a}$

- Recall that a linear equation with zero intercept is $\hat{y} = a_1 x$

$$a_1 = \frac{y^T x}{x^T x}$$

- Therefore in this case:

$$w_{k,a} = \frac{\boldsymbol{u}_a^T \boldsymbol{x}_k}{\boldsymbol{u}_a^T \boldsymbol{u}_a}$$



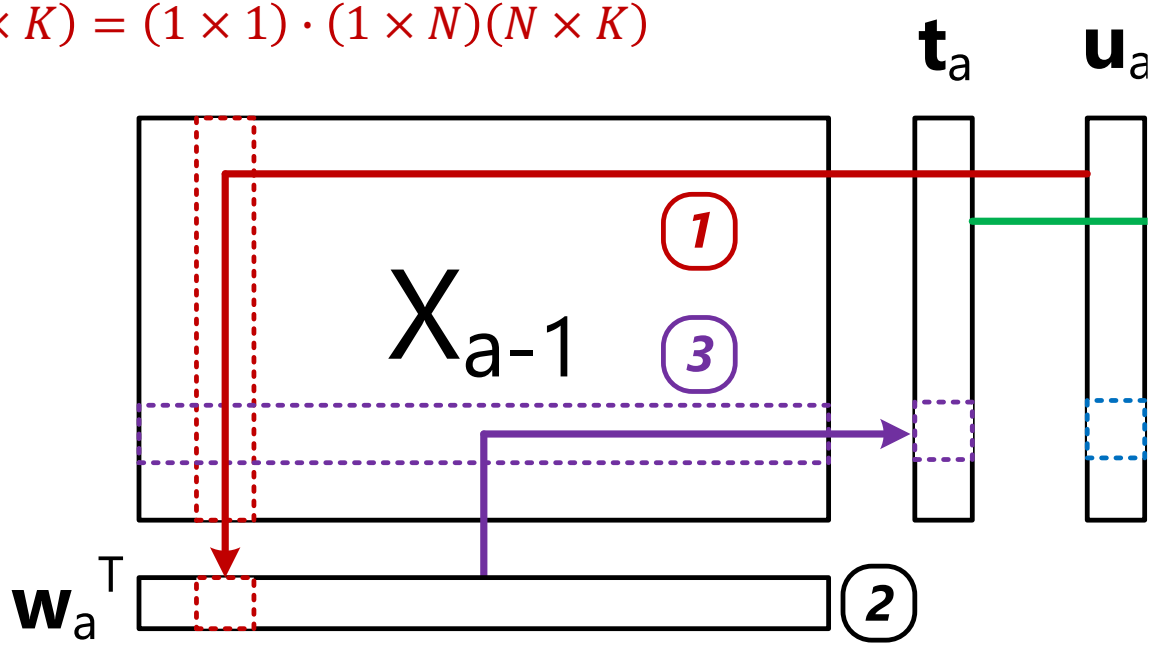
NIPALS for PLS Step By Step

- STEP 2.1 Repeat regression for all columns in X_{a-1}
- Can compute all regressions in one go (HOW!?)

$$\boldsymbol{w}_a^T = \frac{1}{\boldsymbol{u}_a^T \boldsymbol{u}_a} \cdot \boldsymbol{u}_a^T X_{a-1}$$

$$(1 \times K) = (1 \times 1) \cdot (1 \times N)(N \times K)$$

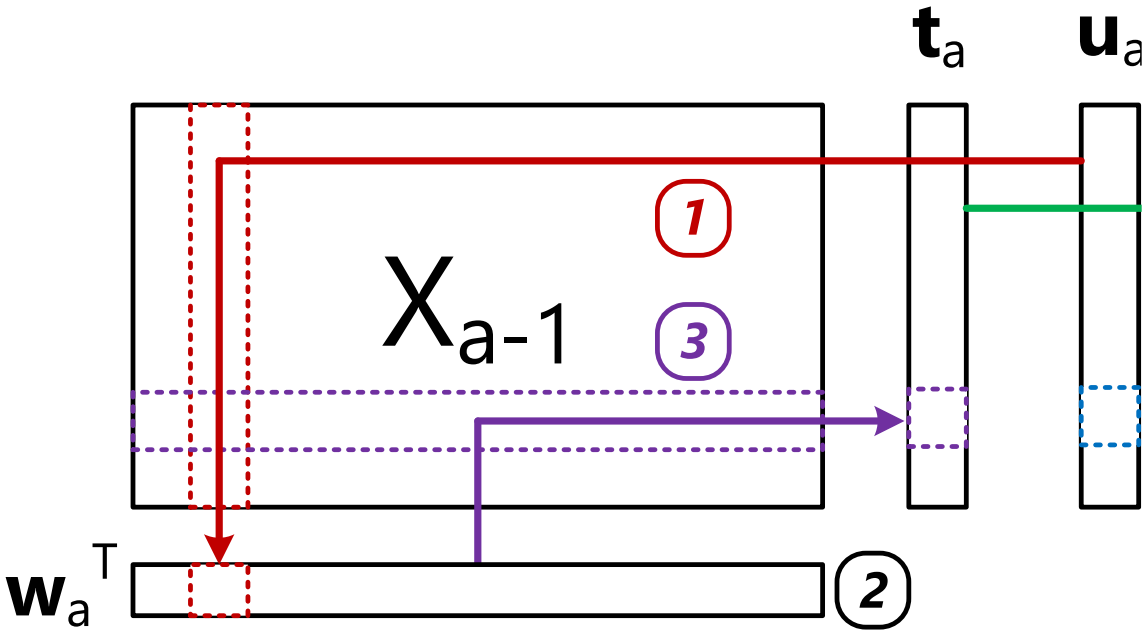
- $\boldsymbol{u}_a \in \mathbb{R}^{N \times 1}$
- $X_{a-1} \in \mathbb{R}^{N \times K}$
- $\boldsymbol{w}_a \in \mathbb{R}^{K \times 1}$



NIPALS for PLS Step By Step

- STEP 2.2 Normalize the weights ------ I'll gently point out the nomenclature difference here
  - $\boldsymbol{w}_a$ will not have unit length
  - Rescale to magnitude of 1.0:

$$\boldsymbol{w}_a = \frac{\boldsymbol{w}_a}{\|\boldsymbol{w}_a\|}$$



NIPALS for PLS Step By Step

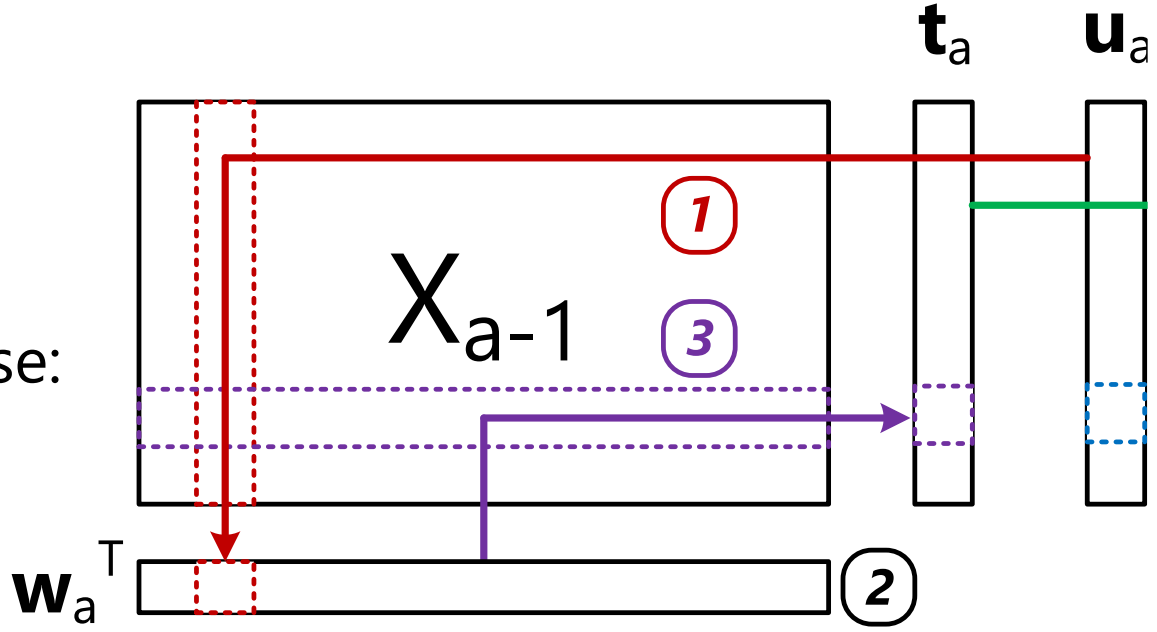
- STEP 2.3 Regress each row of X_{a-1} $(\boldsymbol{x}_n)$ onto $\boldsymbol{w}_a^T$
  - Perform a LS regression of $\boldsymbol{x}_n$ onto $\boldsymbol{w}_a^T$ *(regress y onto x)*
  - Store the regression coefficient as $t_{n,a}$

- Recall that a linear equation with zero intercept is $\hat{y} = a_1 x$

$$a_1 = \frac{y^T x}{x^T x}$$

- Therefore in this case:

$$t_{n,a} = \frac{\boldsymbol{w}_a^T \boldsymbol{x}_n}{\boldsymbol{w}_a^T \boldsymbol{w}_a}$$



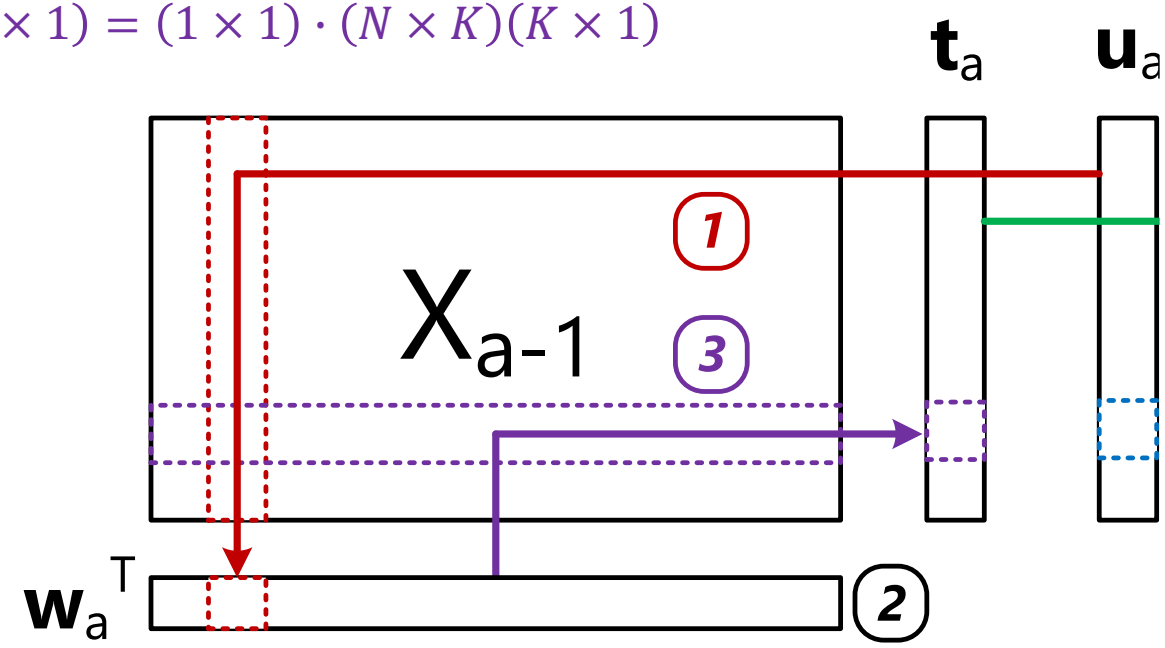
NIPALS for PLS Step By Step

- STEP 2.3 Repeat regression for all rows in X_{a-1}
- Can compute all regressions in one go

$$\boldsymbol{t}_a = \frac{1}{\boldsymbol{w}_a^T \boldsymbol{w}_a} \cdot X_{a-1} \boldsymbol{w}_a$$

$$(N \times 1) = (1 \times 1) \cdot (N \times K)(K \times 1)$$

- $\boldsymbol{t}_a \in \mathbb{R}^{N \times 1}$
- $X_{a-1} \in \mathbb{R}^{N \times K}$
- $\boldsymbol{w}_a \in \mathbb{R}^{K \times 1}$



NIPALS for PLS Step By Step

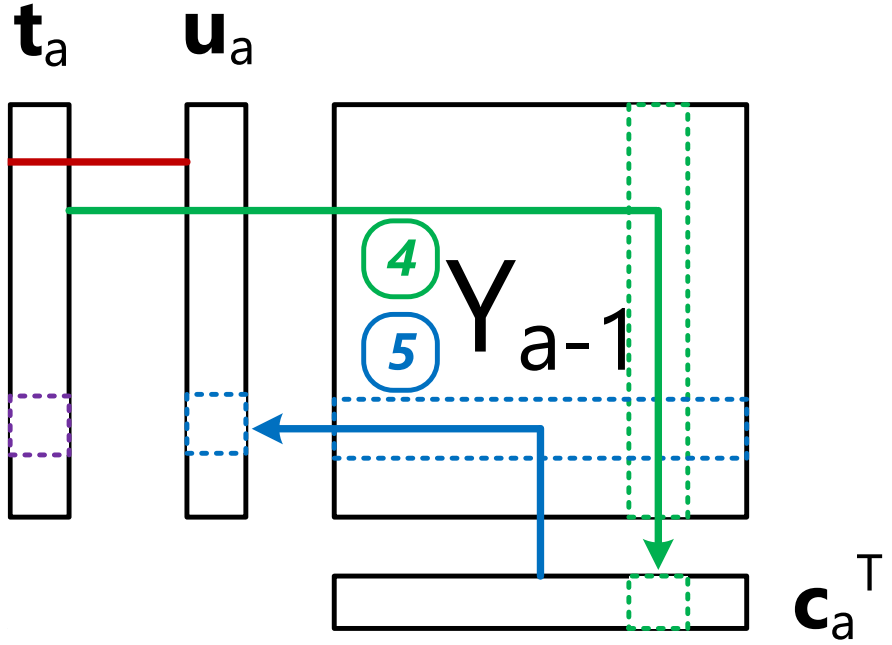
- STEP 2.4 Regress each column of Y_{a-1} ($\boldsymbol{y}_m$) onto $\boldsymbol{t}_a$
  - Perform a LS regression of $\boldsymbol{y}_m$ onto $\boldsymbol{t}_a^T$ *(regress y onto x)*
  - Store the regression coefficient as $c_{m,a}$

- Recall that a linear equation with zero intercept is $\hat{y} = a_1 x$

$$a_1 = \frac{y^T x}{x^T x}$$

- Therefore in this case:

$$c_{m,a} = \frac{\boldsymbol{t}_a^T \boldsymbol{y}_m}{\boldsymbol{t}_a^T \boldsymbol{t}_a}$$



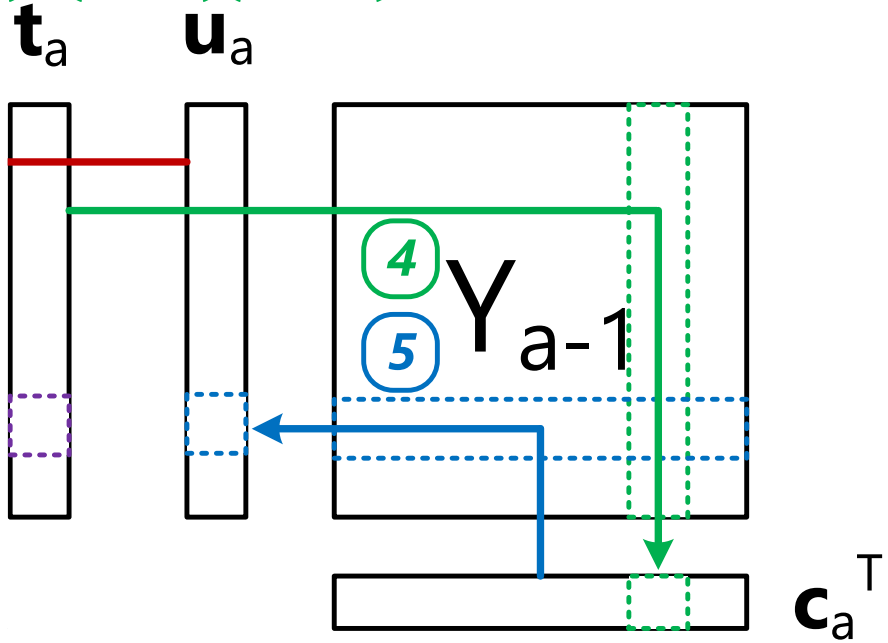
NIPALS for PLS Step By Step

- STEP 2.4 Repeat regression for all columns in Y_{a-1}
- Can compute all regressions in one go

$$\boldsymbol{c}_a^T = \frac{1}{\boldsymbol{t}_a^T \boldsymbol{t}_a} \cdot \boldsymbol{t}_a^T Y_{a-1}$$

$$(1 \times M) = (1 \times 1) \cdot (1 \times N)(N \times M)$$

- $\boldsymbol{t}_a \in \mathbb{R}^{N \times 1}$
- $Y_{a-1} \in \mathbb{R}^{N \times M}$
- $\boldsymbol{c}_a \in \mathbb{R}^{M \times 1}$



NIPALS for PLS Step By Step

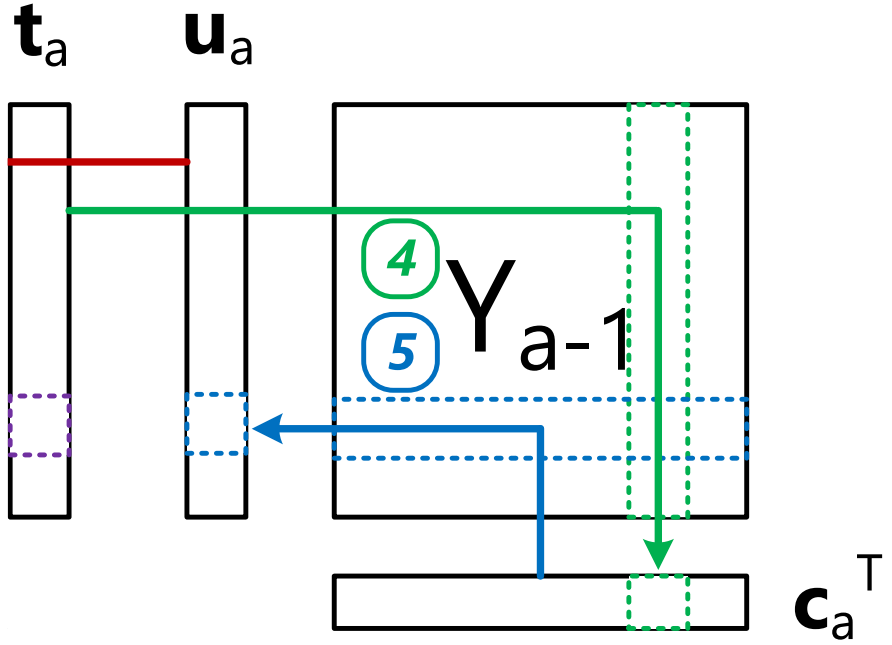
- STEP 2.5 Regress each row of Y_{a-1} $(\boldsymbol{y}_n)$ onto $\boldsymbol{c}_a$
  - Perform a LS regression of $\boldsymbol{y}_n$ onto $\boldsymbol{c}_a$ *(regress y onto x)*
  - Store the regression coefficient as $u_{n,a}$

- Recall that a linear equation with zero intercept is $\hat{y} = a_1 x$

$$a_1 = \frac{y^T x}{x^T x}$$

- Therefore in this case:

$$u_{n,a} = \frac{\boldsymbol{c}_a^T \boldsymbol{y}_n}{\boldsymbol{c}_a^T \boldsymbol{c}_a}$$



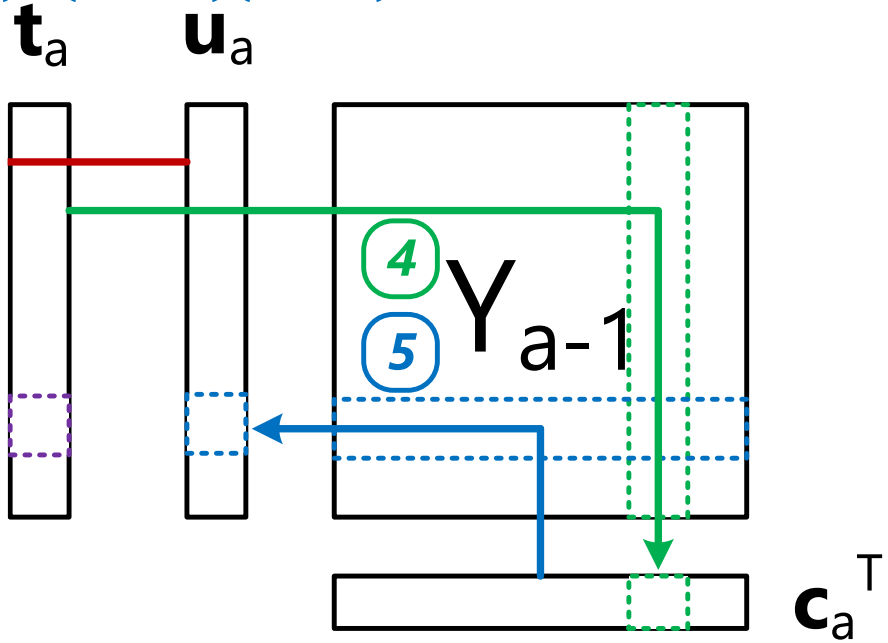
NIPALS for PLS Step By Step

- STEP 2.5 Repeat regression for all rows in Y_{a-1}
- Can compute all regressions in one go

$$\boldsymbol{u}_a = \frac{1}{\boldsymbol{c}_a^T \boldsymbol{c}_a} \cdot Y_{a-1} \boldsymbol{c}_a$$

$$(N \times 1) = (1 \times 1) \cdot (N \times M)(M \times 1)$$

- $\boldsymbol{u}_a \in \mathbb{R}^{N \times 1}$
- $Y_{a-1} \in \mathbb{R}^{N \times M}$
- $\boldsymbol{c}_a \in \mathbb{R}^{M \times 1}$



NIPALS Step By Step

- CHECK FOR CONVERGENCE
  - Compare $\boldsymbol{u}_a$ to $\boldsymbol{u}_a$ from previous iteration
  - Stop if $\Delta\boldsymbol{u}_a \leq \sqrt{\epsilon} \approx 10^{-8}$
  - Change could mean $\left\|\boldsymbol{u}_a^i - \boldsymbol{u}_a^{i-1}\right\| \leq \sqrt{\epsilon}$ [absolute]
  - Change could mean $\frac{\left\|\boldsymbol{u}_a^i - \boldsymbol{u}_a^{i-1}\right\|}{\left\|\boldsymbol{u}_a^{i-1}\right\|} \leq \sqrt{\epsilon}$ [relative]
  - ALSO probably want an iterations limit in case (300 is good)

- AT CONVERGENCE
  - $\boldsymbol{t}_a$, $\boldsymbol{w}_a$, $\boldsymbol{u}_a$ and $\boldsymbol{c}_a$ are the a^{th} component
  - Store as columns in T, W, U, and C, respectively

OBSERVATION

- Something **sneaky** has happened here
  - Since we never regressed $\boldsymbol{t}_a$ onto X_{a-1}, we actually never computed the **loadings** of X_{a-1} as related to $\boldsymbol{p}_a$

- WORKSHOP: How then do we find the **residuals** E_a of X_{a-1}? It is NOT fair to use w_a (our weights) [WHY?]

NIPALS Step By Step

- Step 3 Deflation
- 3.1. Compute loadings $\boldsymbol{p}_a = \frac{1}{\boldsymbol{t}_a^t \boldsymbol{t}_a} X_{a-1}^T \boldsymbol{t}_a$
  - Regressing columns of X_{a-1} on converged scores $\boldsymbol{t}_a$
- 3.2. Remove predicted variance in X_{a-1} and Y_{a-1}
  - Deflate X_{a-1}:
    - $E_a = X_{a-1} - \hat{X}_{a-1}$ $\quad E_a = X_{a-1} - \boldsymbol{t}_a \boldsymbol{p}_a^T$
    - Set new deflated X $\quad X_a = E_a$
  - Deflate Y_{a-1}:
    - $F_a = Y_{a-1} - \hat{Y}_{a-1}$ $\quad F_a = Y_{a-1} - \boldsymbol{t}_a \boldsymbol{c}_a^T$
    - Set new deflated Y $\quad Y_a = F_a$

Yes it actually is $\boldsymbol{t}_a$, that is not a typo

Interpreting the Weights

- Like in PCA, NIPALS computes scores based on deflated matrices

- However, UNLIKE in PCA, those scores are (likely) not the scores that explain the greatest variance
  - Recall we are trading-off the explanations of X, Y and $r\{X, Y\}$

- Therefore the weights $\boldsymbol{w}_a$ we calculate can actually only be applied to the **deflated matrix** X_{a-1} used during the NIPALS algorithm
  - We could do this in PCA as well, but it would not change anything

Interpreting the Weights

- So, ANY scores $\boldsymbol{t}_a$ must be calculated using X_{a-1}
  - $\boldsymbol{t}_1 = X_0\boldsymbol{w}_1$ [Nothing spiced here]
  - $\boldsymbol{t}_2 = X_1\boldsymbol{w}_2$ [Note we have to use X_1 here]

- The weights $\boldsymbol{w}_2$ really only have meaning for the DELFATED matrix X_1. This is annoying for two reasons:
  1. It is hard to interpret because $\boldsymbol{w}_2$ and $\boldsymbol{w}_1$ can be directly compared like $\boldsymbol{p}_2$ and $\boldsymbol{p}_1$ could, for example
  2. To calculate ALL scores, we are NOT allowed to use $T = XW$ since X means something "different" for each column of W

- We would likely prefer a variant of W that has the same interpretive meaning as P from PCA

Interpreting the Weights: $W^\star$

- Consider some way to get $\boldsymbol{t}_a$ using X_0 only. We call it $\boldsymbol{w}^\star$
  - $\boldsymbol{t}_1 = X_0\boldsymbol{w}_1^\star$ [Same as before]
  - $\boldsymbol{t}_2 = X_0\boldsymbol{w}_2^\star$ [Note we have to use X_0 here]
  - $\boldsymbol{t}_A = X_0\boldsymbol{w}_A^\star$ [For all components]

- We can actually obtain this collection of $\boldsymbol{w}_a^\star$ values by exploiting the loadings we calculated:

$$W^\star = W(P^T W)^{-1}$$

$$(N \times A) = (N \times A) \cdot ([A \times K][K \times A])$$

- So we get $T = XW^\star$ in one go (note X is just X_0 for us)
  - This is a cleaner representation of the relationships in X
  - Note that $w_1^\star = w_1$ BUT $w_a^\star \neq w_a \; \forall \; a > 1$

DETOUR: Why do we use $\hat{X} = \boldsymbol{t}\,\boldsymbol{p}^T$?

- You might be a little freaked out that we want to calculate $\hat{X}$ based on the scores in t only
  - Instead, you might realize that we regressed the columns of X_{a-1} onto $\boldsymbol{w}_a$ to get $\boldsymbol{u}_a$
  - It therefore might make sense to estimate $\hat{X}_{a-1} = \boldsymbol{u}_a \boldsymbol{w}_a^T$

- This leads to a **CRITICAL** point about PLS: Our objective is to train the PLS model so we can *predict* Y in the future… Which means… We don't have Y
  - So, we **can't** predict the Y-space scores to get $U = YC$
  - If we don't have U, we **can't** get $\hat{X} = UW^T$
  - INSTEAD, we would like to have some way to estimate $\hat{X}$ using T only, since with a new data point we can get $T_{new} = X_{new}W^\star$
  - This is **important** because we need $\hat{X}$ to estimate SPE
  - Result: We actually don't end up using U ever again…

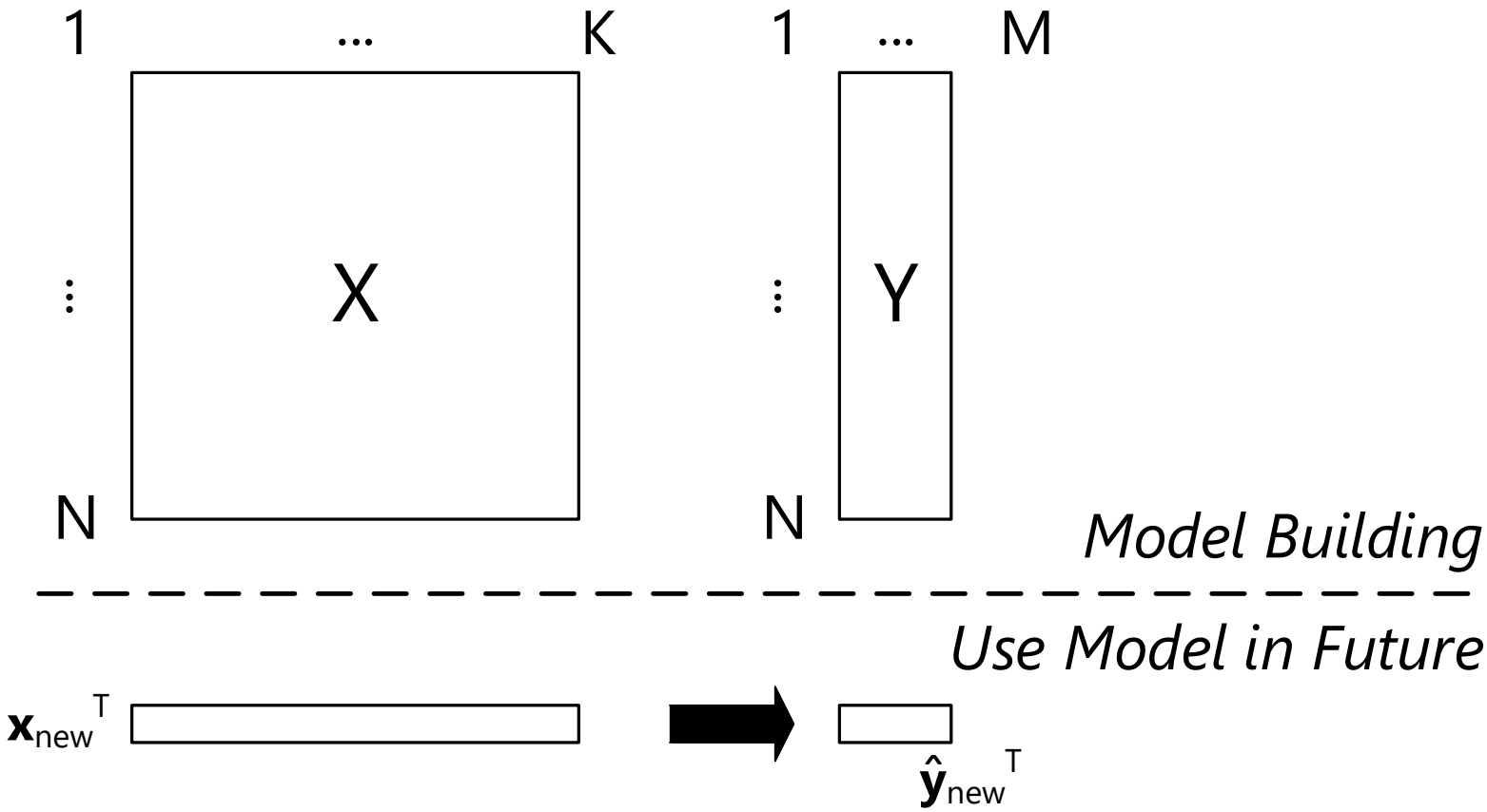
McMaster University ENGINEERING

Using PLS on New Data

Y predict Y if you are not going to predict Y?

Using PLS on New Data

- Our objective here is to train the model using KNOWN values of Y so we can predict those outcomes later
  - To do this, we sacrifice explanation of X more than PCA

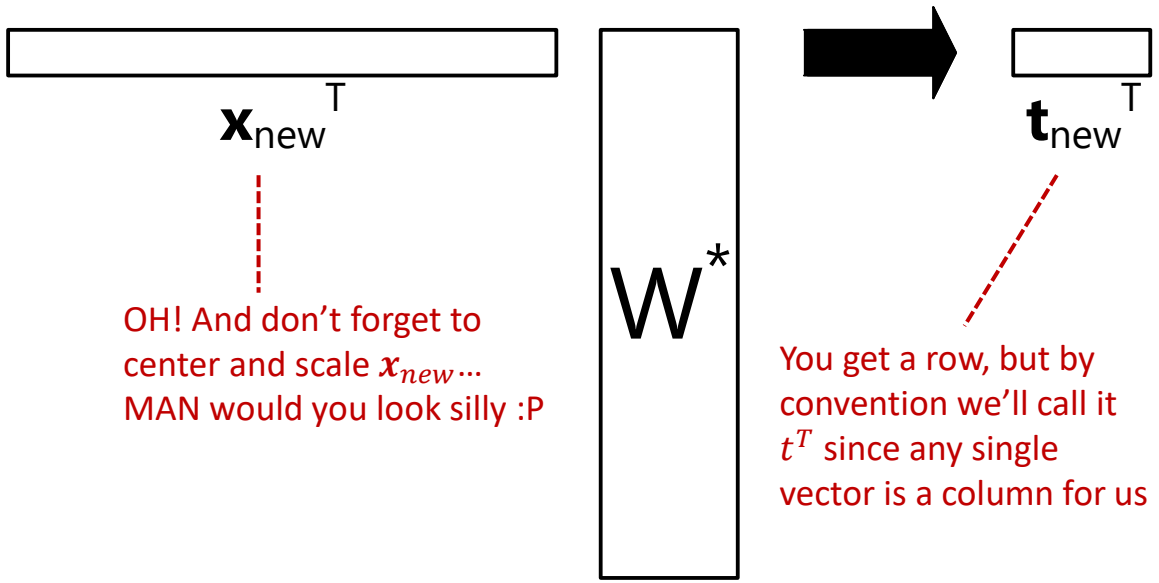


Using PLS on New Data

FIRST: Get Scores

$$t_{1,new} = \boldsymbol{x}_{new}^T \boldsymbol{w}_1$$
$$\boldsymbol{x}_{new}^T = \boldsymbol{x}_{new}^T - t_{1,new}\, \boldsymbol{p}_1^T$$
$$t_{2,new} = \boldsymbol{x}_{new}^T \boldsymbol{w}_2$$
$$\boldsymbol{x}_{new}^T = \boldsymbol{x}_{new}^T - t_{2,new} \boldsymbol{p}_2^T$$

…



OH! And don't forget to center and scale $\boldsymbol{x}_{new}$… MAN would you look silly :P

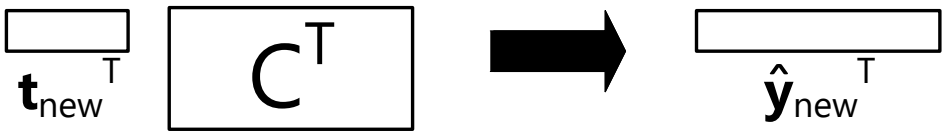
You get a row, but by convention we'll call it t^T since any single vector is a column for us

OR USE $W^\star$ to get whole vector $\boldsymbol{t}_{new}$ without deflation:

$$\boldsymbol{t}_{new}^T = \boldsymbol{x}_{new}^T W^\star$$

Using PLS on New Data

NEXT: Get Y



$$\hat{\boldsymbol{y}}_{new} = \boldsymbol{t}_{new}^T \boldsymbol{C}^T$$

ALSO remember to UNCENTER and UNSCALE your $\boldsymbol{y}_{new}$ lest you look silly ☺

- Note that you **CAN** skip the middle step and go:

$$\hat{y}_{new} = x_{new}^T W^\star C^T$$

- You should NEVER implement PLS this way
  - We WANT the scores to calculate our model stats (T^2 and SPE)

Cross-Validation

- One big advantage of predicting $\hat{\boldsymbol{y}}$ is that we can use it to determine the optimal number of components!
  - You may recall we did this in PCA using Q^2 and PRESS
  - The procedure is the same, but we will do it for Y instead!

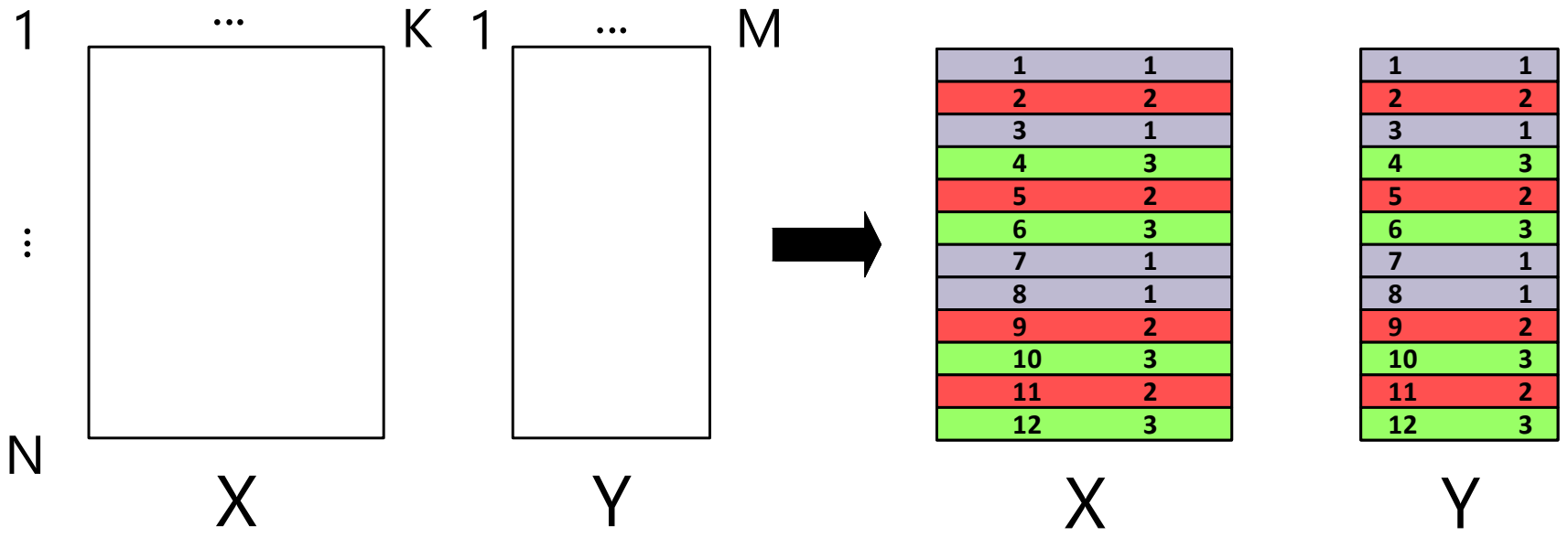
- Recall for training data: $R^2 = 1 - \frac{\mathcal{V}(F_A)}{\mathcal{V}(Y)}$
- DEFINE for testing data: $Q^2 = 1 - \frac{\mathcal{V}(F_A \text{ predicted})}{\mathcal{V}(Y)}$

Use the Y residuals and variance here

- $\mathcal{V}(E_A \text{ predicted})$ is known as the **PRESS**

Cross-Validation

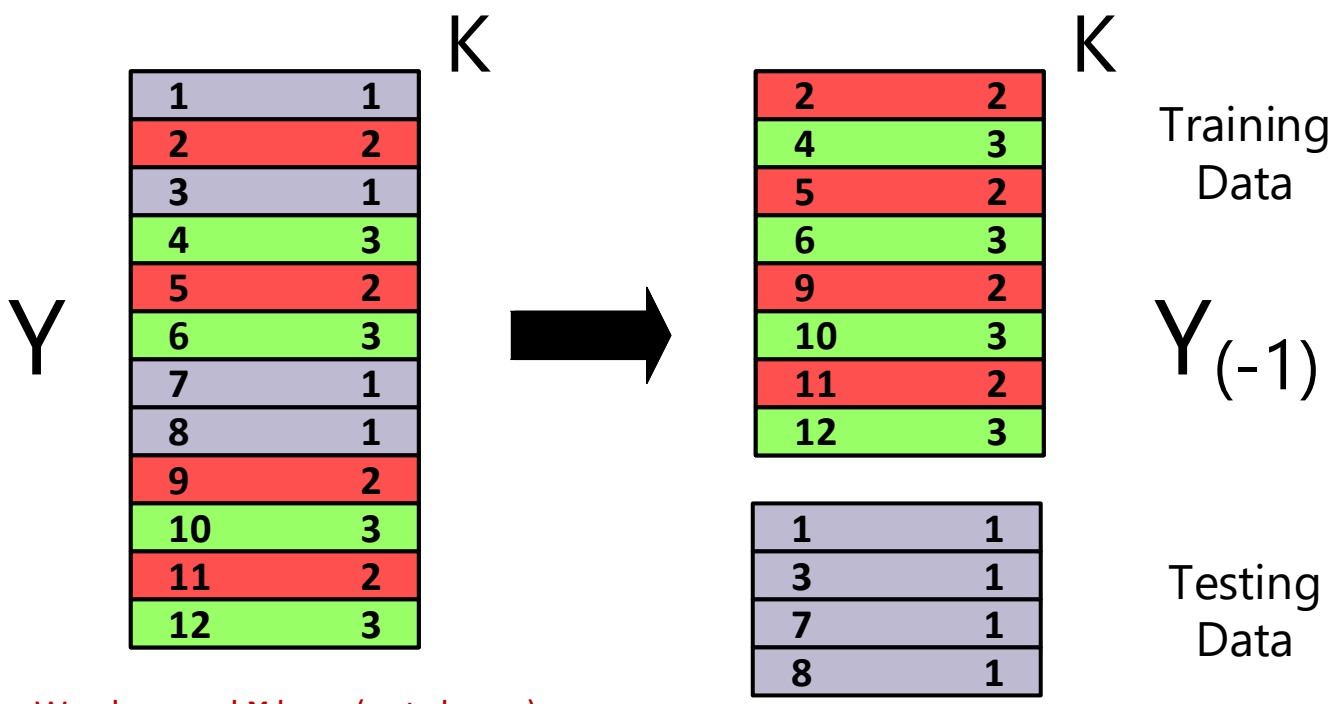
- Step 1: Split rows of X AND Y into **G** groups (3 here)
  - Typically **G** is 7 in software packages
  - Can be random or ordered (random in this example)
  - Note that this can depend on time relevance!



Cross-Validation

- Step 2.1: Fit PLS model
  - Use $\mathbf{X}_{(-1)}$ and $\mathbf{Y}_{(-1)}$ for fitting
  - Use $\mathbf{X}_{(1)}$ and $\mathbf{Y}_{(1)}$ for testing
- Compute $F_{(1)} = Y_{(1)} - \hat{Y}_{(1)}$

Important to remember that the main point of PLS is to predict **Y**, so we are using the **Y** residuals here

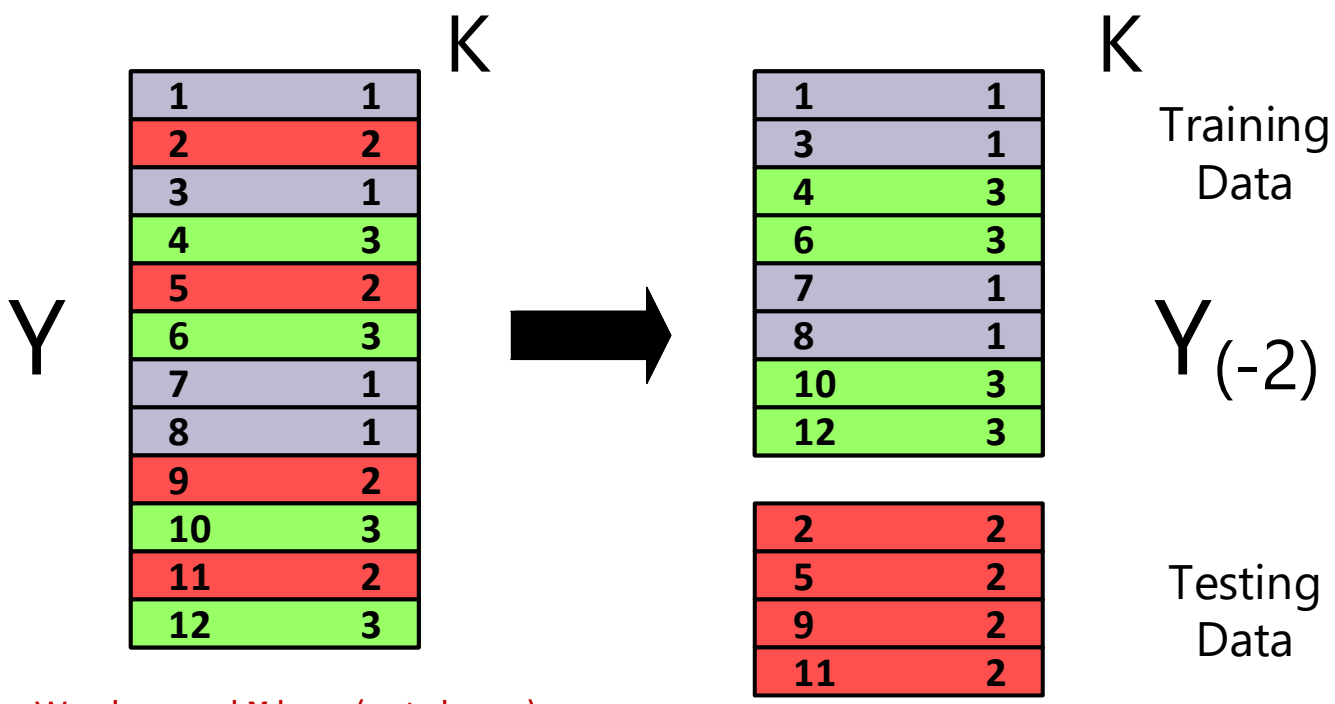


We also need **X** here (not shown)

Cross-Validation

- Step 2.2: Fit PLS model
  - Use $\mathbf{X}_{(-2)}$ and $\mathbf{Y}_{(-2)}$ for fitting
  - Use $\mathbf{X}_{(2)}$ and $\mathbf{Y}_{(2)}$ for testing
- Compute $F_{(2)} = Y_{(2)} - \hat{Y}_{(2)}$

Important to remember that the main point of PLS is to predict **Y**, so we are using the **Y** residuals here

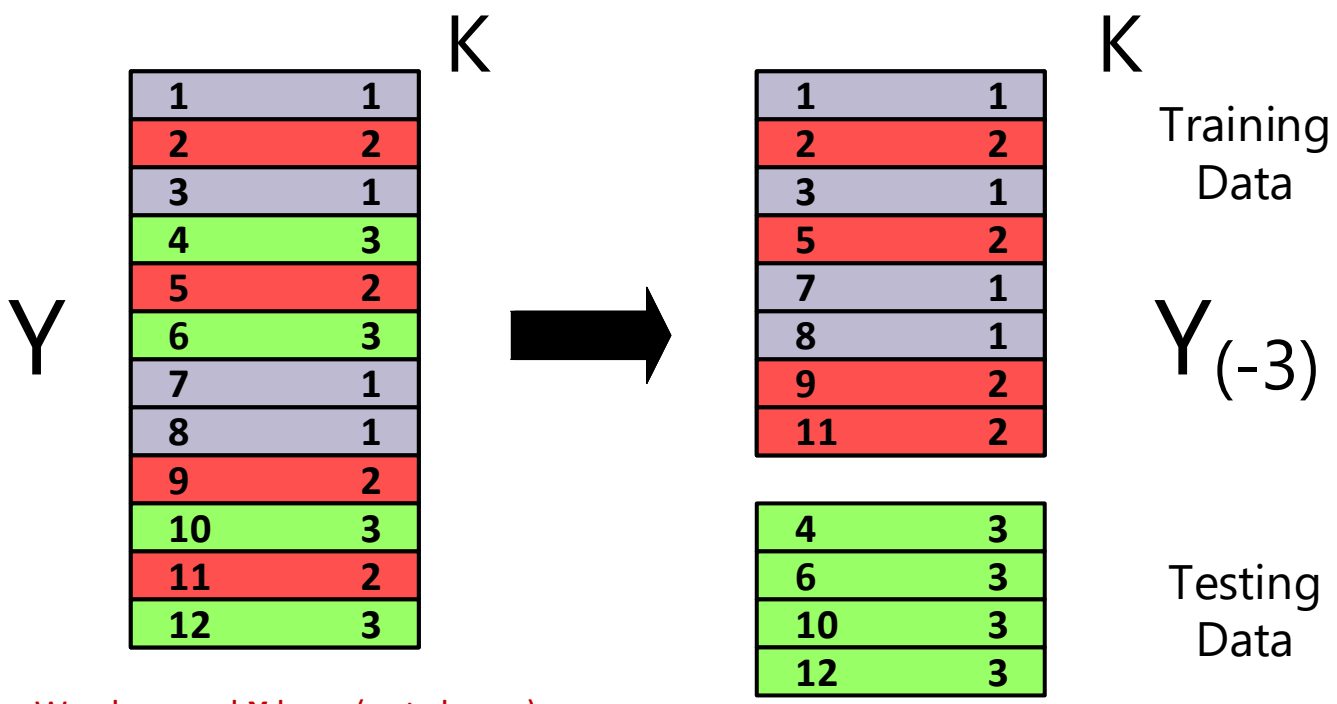


We also need **X** here (not shown)

Cross-Validation

- Step 2.3: Fit PLS model
  - Use $\mathbf{X}_{(-3)}$ and $\mathbf{Y}_{(-3)}$ for fitting
  - Use $\mathbf{X}_{(3)}$ and $\mathbf{Y}_{(3)}$ for testing
- Compute $F_{(3)} = Y_{(3)} - \hat{Y}_{(3)}$

Important to remember that the main point of PLS is to predict **Y**, so we are using the **Y** residuals here



We also need **X** here (not shown)

Cross-Validation

- Step 2.G: Fit PLS model
  - Use $\mathbf{X}_{(-G)}$ and $\mathbf{Y}_{(-G)}$ for fitting
  - Use $\mathbf{X}_{(G)}$ and $\mathbf{Y}_{(G)}$ for testing
- Compute $F_{(G)} = Y_{(G)} - \hat{Y}_{(G)}$

Juuuust pretend I have a nice "Gth" group separated out here…

Q^2 Calculations and Interpretation

- Step 3.1: Calculate **PRESS**
  - $\text{PRESS} = \text{ssq}(F_{(1)}) + \text{ssq}(F_{(2)}) + \cdots + \text{ssq}(F_{(G)})$
- Step 3.2: Calculate Q^2
  - $Q^2 = 1 - \frac{\mathcal{V}(\text{predeicted } F_A)}{\mathcal{V}(Y)} \Rightarrow Q^2 = 1 - \frac{\text{PRESS}}{\mathcal{V}(Y)}$
- Interpretations of Q^2
  - Interpreted the same way as R^2 (higher the better, 1 is best)
  - You should find that $Q^2 \leq R^2$ (unless you are very lucky)
  - If $Q^2 \approx R^2$ for an a^{th} component, the component is **useful**
  - If Q^2 is very small, likely fitting noise
  - Q^2 for an a^{th} components CAN be negative (why?)
  - Q^2_k can be calculated for specific variable k

McMaster University ENGINEERING

EXAMPLE DATA SET

Polymer Reaction Process

Example: Polymers Reactor

- Consider the data set available on A2L called `LDPE.csv`

- This data set contains 14 columns in **X** that describe two tubular polymerization reactors in series:
  - Tin: inlet temperature to zone 1 of the reactor [K]
  - Tmax1: maximum temperature along zone 1 [K]
  - Tout1: outlet temperature from zone 1 [K]
  - Tmax2: maximum temperature along zone 2 [K]
  - Tout2: outlet temperature from zone 2 [K]
  - Tcin1: temperature of inlet coolant to zone [K]
  - Tcin2: temperature of inlet coolant to zone 2 [K]
  - z1: percentage along zone 1 where Tmax1 occurs [%]
  - z2: percentage along zone 2 where Tmax2 occurs [%]
  - Fi1: flow rate of initiators to zone 1 [g/s]
  - Fi2: flow rate of initiators to zone 2 [g/s]
  - Fs1: flow rate of solvent to zone 1 [% of ethylene]
  - Fs2: flow rate of solvent to zone 2 [% of ethylene]
  - Press: pressure in the reactor [atm]

Example: Polymers Reactor

- Consider the data set available on A2L called `LDPE.csv`
- This data set also contains 5 columns in **Y** that describe several quality metrics of the product out of the reactors:
  - Conv: quality variable: cumulative conversion
  - Mn: quality variable: number average molecular weight
  - Mw: quality variable: weight average molecular weight
  - LCB: quality variable: long chain branching per 1000 C atoms
  - SCB: quality variable: short chain branching per 1000 C atoms
- We would like to build a PLS model on this data using TWO components

McMaster University ENGINEERING

Analyzing Data with PLS Plots

Y predict Y if you are not going to predict Y?

PLS Plots (Explored in Demo)

- Score plots
  - $\boldsymbol{t}$ and $\boldsymbol{u}$ show relationships between rows
- Weight plots
  - $\boldsymbol{w}$ shows **deflated** relationship between columns in X
  - $\boldsymbol{w}^{\star}$ shows **un-deflated** relationship between columns in X (preferred)
- Loadings plots
  - $\boldsymbol{c}$ shows relationship between columns in Y
- Weight AND loadings plots
  - Super-imposed $\boldsymbol{w}^{\star}$ and $\boldsymbol{c}$ plots show relationship between X and Y
- Modeling statistics plots
  - SPE (for X and Y spaces) to know if outlier off model plane
  - T^2 to know if outlier on model plane
  - R^2 plots for overall, by variable in X or Y, *etc*.
- Coefficient plot (described next)

Coefficient Plots

- Recall I mentioned a "quick" way to get $\hat{y}_{new}$:

$$\hat{\boldsymbol{y}}_{new} = \boldsymbol{x}_{new}^T W^\star C^T$$

- This actually kind of looks like a linear regression, no?

$$\hat{\boldsymbol{y}}_{new} = \boldsymbol{x}_{new}^T \boldsymbol{\beta}$$

- $\boldsymbol{\beta} \in \mathbb{R}^{(K \times N)}$ is a collection of "importance" of each column in X as it relates to each column in Y
  - For one column in $\boldsymbol{\beta}$, we would just end up with the MLR equation for regressing $\boldsymbol{y}$ onto X!
  - However, it is critical to note that this is **DIFFERENT** than MLR

Example Coefficient Plot

- Consider our polymer reaction process
  - Below plot relates X to conversion ($\boldsymbol{y}$) with 5 components
  - $\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_K x_K \quad \Leftrightarrow \quad \hat{\boldsymbol{y}} = \boldsymbol{x}^T \boldsymbol{\beta}$
  - **IMPORTANT**: this is good for visualization but you should **NEVER** implement PLS this way

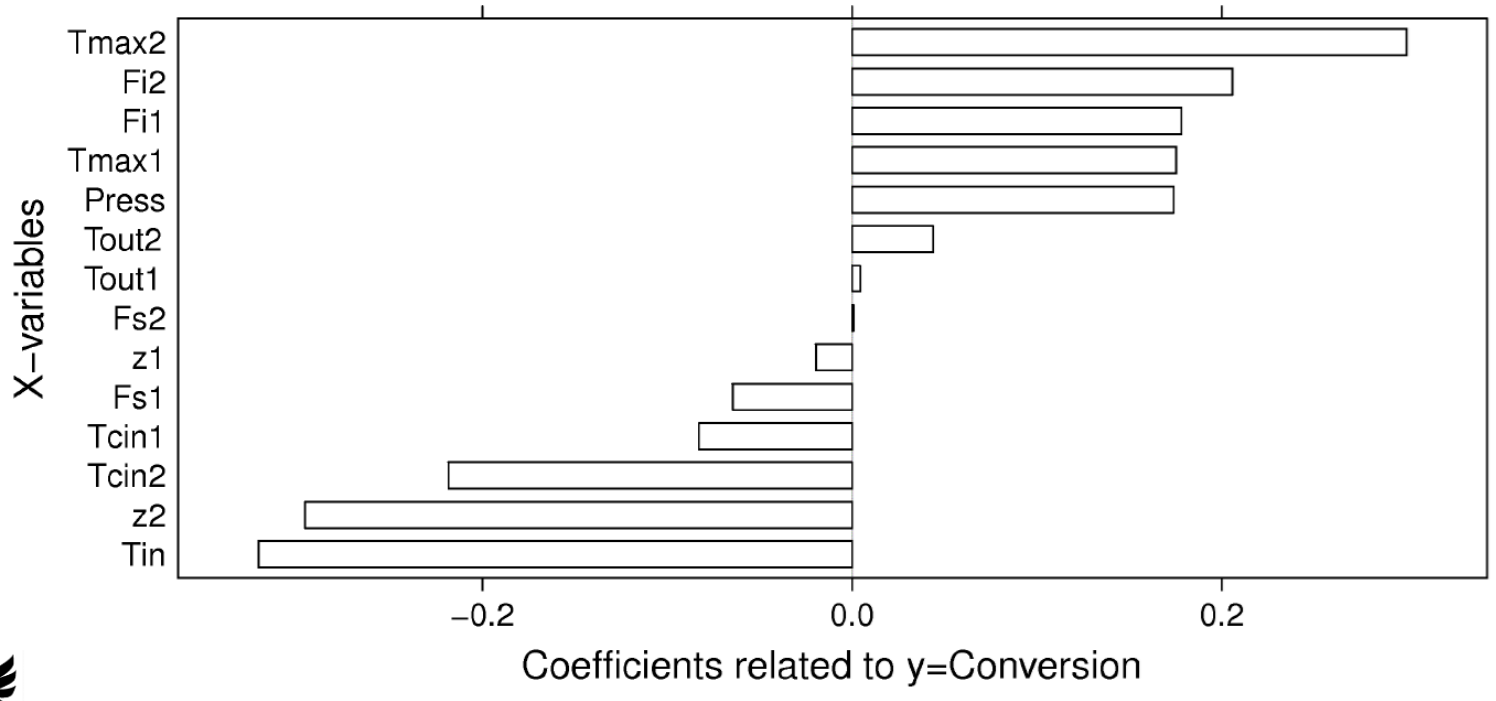


Image courtesy of ConnectMV

Key Points

- PLS is an incredible modeling tool relating X to Y
  - Simultaneously explains variance and relationships
- NIPALS determines orthogonal weights and loadings that meet the objective of maximizing $\mathcal{C}(\boldsymbol{t}, \boldsymbol{u})$
- We don't want to rely on scores in the Y space in future
  - So, we calculate loadings $\boldsymbol{p}$ from converged scores $\boldsymbol{t}$ (fit using Y) so we can predict $\boldsymbol{t}$ on new data
  - We also deflate X and Y based off of the scores in $\boldsymbol{t}$
- The weights $\boldsymbol{w}$ are therefore annoying to analyze
  - Rely on deflated matrix X_a and are hard to interpret
  - Modified weights $\boldsymbol{w}^\star$ represent "projections" of weights $\boldsymbol{w}$ onto X-space loadings $\boldsymbol{p}$
- Cross-validation is possible (just like PCA) for PLS
  - Want to add enough components for Q^2 to continue improving
- Many plots can be used to visualize PLS results
  - Visualization of components, data, scores, regression "coefficients" and more!

And Finally…

- We are done with Latent Variable Methods
  - More practice to come in A4!
- We will now move on to machine learning
  - Well, really just ANNs
- Focus for ANN section
  - Derivations
  - Key fundamental understanding of process/results
  - Perspective
  - Applications using built-in software