Chemical Engineering 4H03

Non-Linear **I**terative **PA**rtial **L**east-**S**quares

Jake Nease
McMaster University

Objectives for this Class

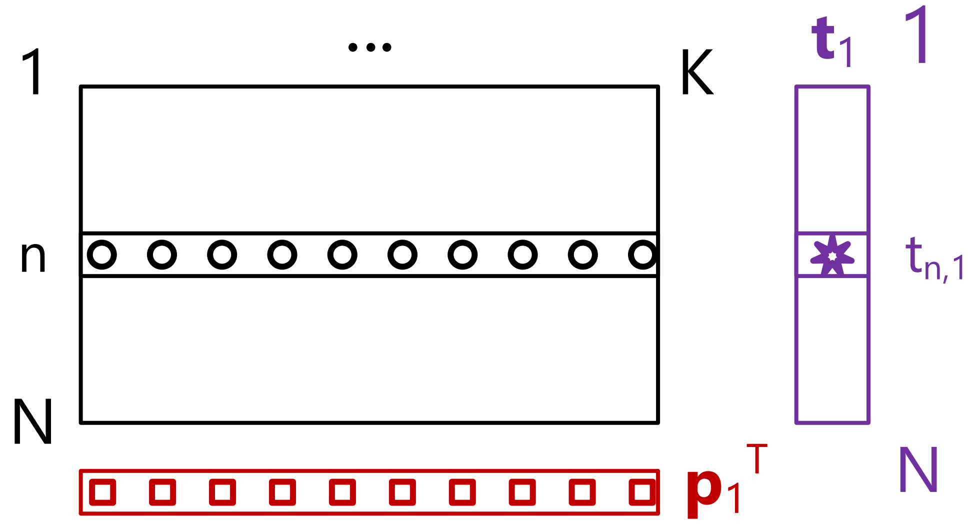
- Where we came from…
  - Computing $\boldsymbol{p}_k$ from Eigenvalue Decomposition
  - Gave us a fundamental understanding of how PCA works
  - Understand that there are **advantages** and **disadvantages**

- Now: is there a more "reliable" way to compute $\boldsymbol{p}_k$?
  - Don't want to compute all Eigenvectors at once
  - Want to handle missing data

- How will we do this?
  1. Introduce NIPALS (Non-Linear Iterative Partial Least-Squares)
  2. See how NIPALS can handle missing data (and what this means for our data matrix **X**)

NIPALS

Now **I**'ll **P**robably **A**ctually **L**ose **S**omeone

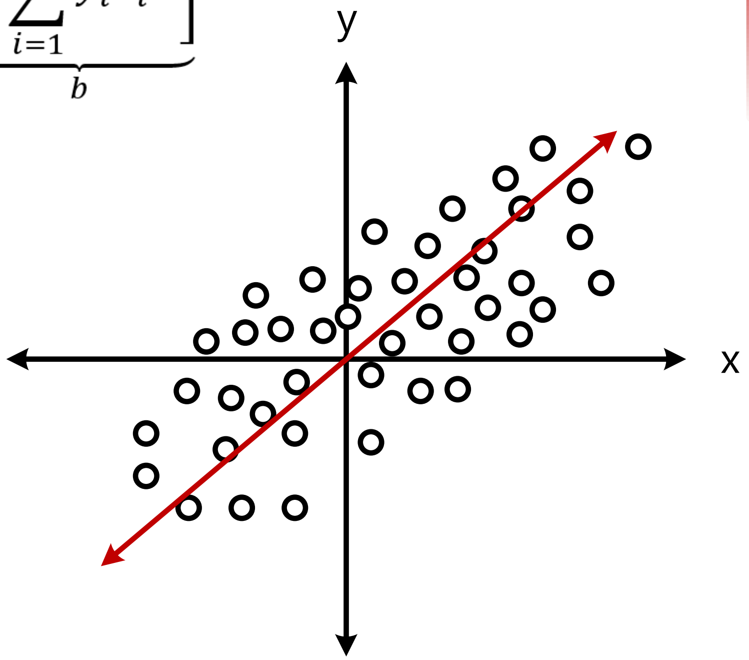


Review: Linear Regression

- You may recall the LSOE that can be solved for the regression coefficients in our regression $\hat{\boldsymbol{y}} = a_0 + a_1\boldsymbol{x}$:

$$\underbrace{\begin{bmatrix} \sum_{i=1}^{N} 1 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} y_i x_i \end{bmatrix}}_{b}$$

- **QUESTION**
  - What is special about the data we are trying to regress in PCA?

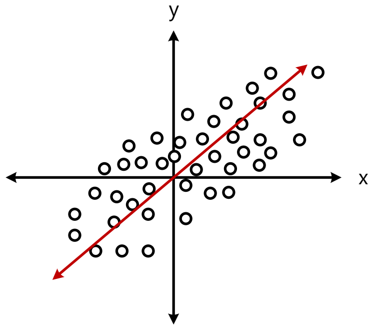


- **ANSWER**
  - It is **centered**, thus $a_0 = 0$

Linear Regression in Vector Form

- Our “system of equations” in that case simply boils down to one equation and one unknown:

$$\underbrace{\begin{bmatrix} \cancel{\sum_{i=1}^{N} 1} & \cancel{\sum_{i=1}^{N} x_i} \\ \cancel{\sum_{i=1}^{N} x_i} & \sum_{i=1}^{N} x_i^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \cancel{a_0} \\ a_1 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \cancel{\sum_{i=1}^{N} y_i} \\ \sum_{i=1}^{N} y_i x_i \end{bmatrix}}_{b} \qquad a_1 = \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} x_i^2}$$



- Fun fact, these sums of the elements in each vector can pretty easily be written as dot-products

$$\sum_{i=1}^{N} y_i x_i = y^T x \qquad\qquad \sum_{i=1}^{N} x_i^2 = x^T x$$

- We can thus collapse our equation for a_1 into:

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

--------- At this point, I should not need to convince you that this expression is identical to the one for a_1 above

NIPALS

- Non-Linear Iterative Partial Least-Squares

- Why study it?
  - Additional insight into what loads and scores mean
  - A different way of looking at orthogonality
  - Handles missing data
  - Does not compute all Eigenvectors (efficient)
  - Used by most popular software packages
    - SIMCA
    - Aspen ProMV
    - Most pre-made Python packages

NIPALS: The Basic Idea

- NIPALS begins with X
  - X is pre-processed via scaling and centering
  - **NOMENCLATURE ADDITION**: X_a is the data set after a components have been fit to it (more on this later)
  - Thus, NIPALS initializes with $X_{a=0} \equiv X_0$ since no components are fit

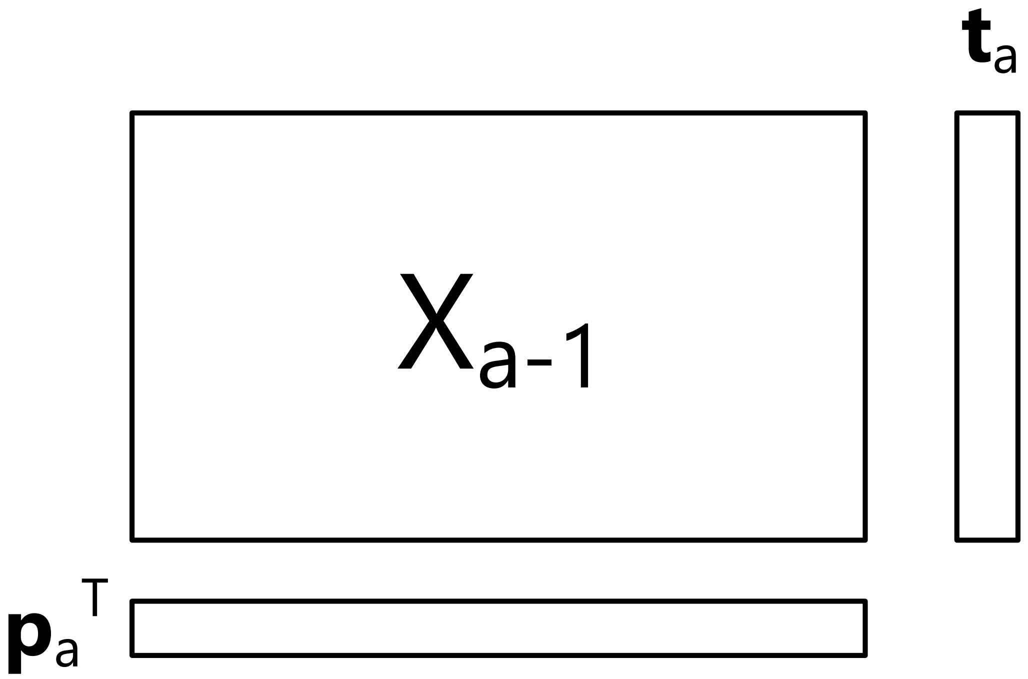
```
FOR $a = 1, 2, \cdots, A$:
    1.  Select an (arbitrary) column as $\boldsymbol{t}_a$
    2.  In a loop until convergence:
        I.   Regress columns from $X_{a-1}$ onto $\boldsymbol{t}_a$ to get $\boldsymbol{p}_a$
        II.  Normalize $\boldsymbol{p}_a$ to unit length
        III. Regress rows from $X_{a-1}$ onto $\boldsymbol{p}_a^T$ to get $\boldsymbol{t}_a$
    3.  Deflate component from $X_{a-1}$ to get $X_a$
END
```

These are much easier to visualize if we use the vector regression notation just discussed

QUESTION: what do you think this means?

NIPALS Step By Step

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



NIPALS Step By Step

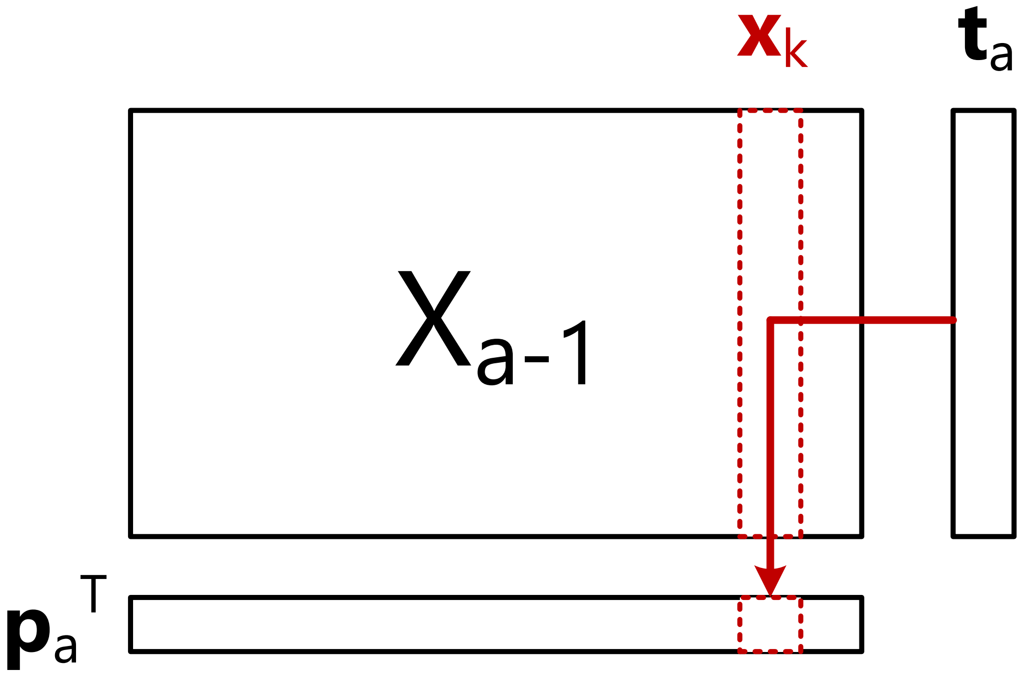
- STEP 2.1 Regress each column of X_{a-1} ($\boldsymbol{x}_k$) onto $\boldsymbol{t}_a$
  - Perform a LS regression of $\boldsymbol{x}_k$ onto $\boldsymbol{t}_a$ *(regress y onto x)*
  - Store the regression coefficient as $p_{k,a}$

- 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:

$$p_{k,a} = \frac{\boldsymbol{t}_a^T \boldsymbol{x}_k}{\boldsymbol{t}_a^T \boldsymbol{t}_a}$$



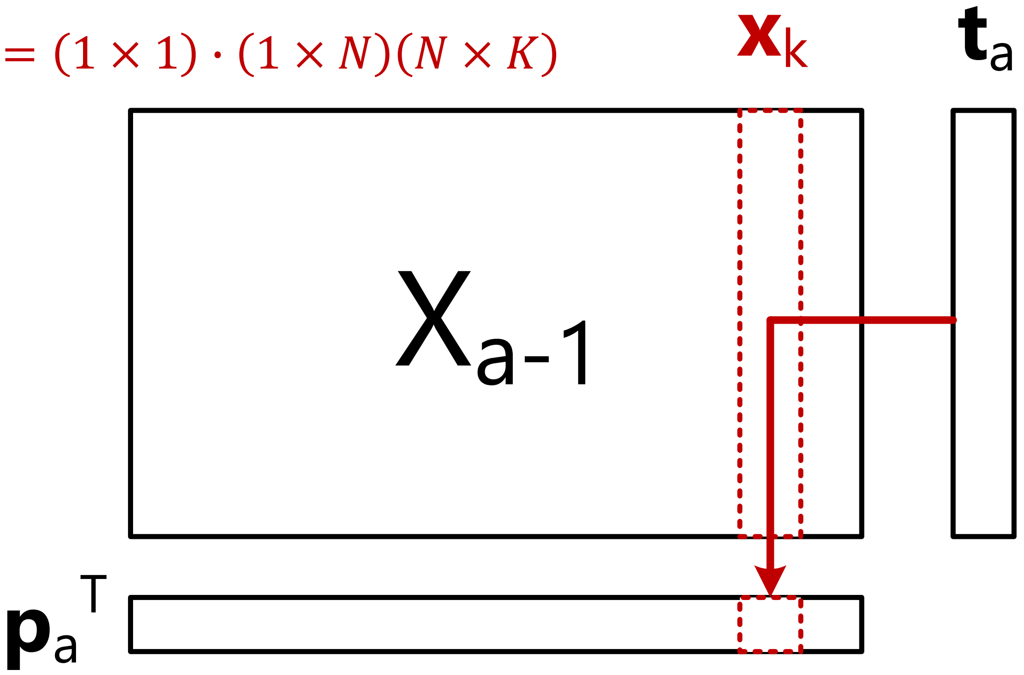
NIPALS Step By Step

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

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

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

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



NIPALS Step By Step

- STEP 2.2 Normalize the loadings
  - $\boldsymbol{p}_a$ will not have unit length (WHY?)
  - Rescale to magnitude of 1.0:

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

Recall that $\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots}$

NIPALS Step By Step

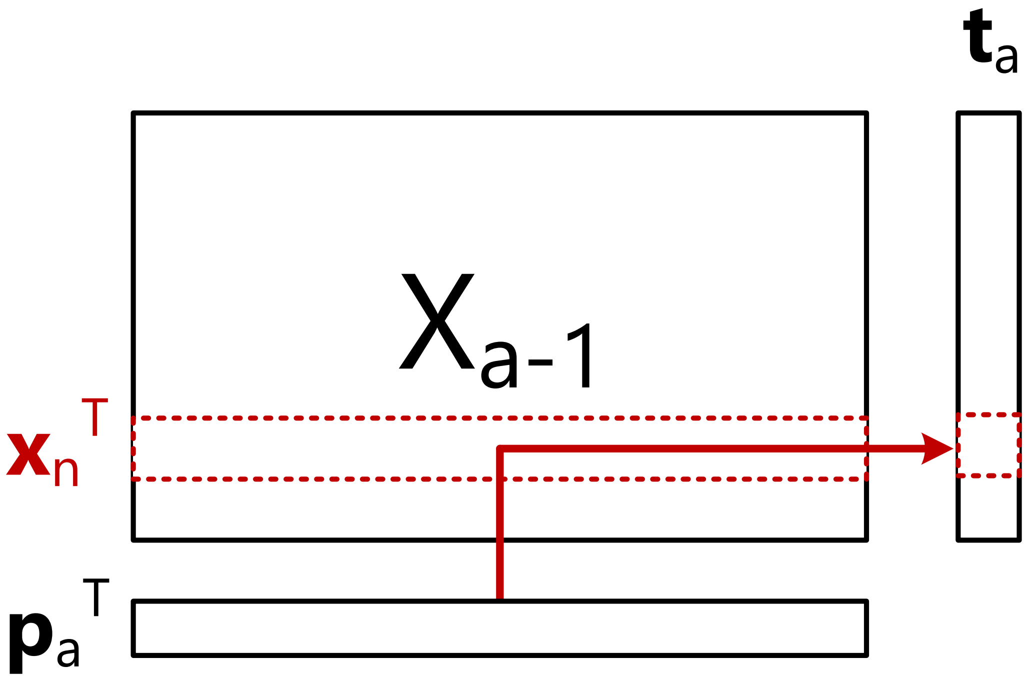
- STEP 2.3 Regress each row of X_{a-1} ($\boldsymbol{x}_n$) onto $\boldsymbol{p}_a^T$
  - Perform a LS regression of $\boldsymbol{x}_n$ onto $\boldsymbol{p}_a^T$ *(regress y onto x)*
  - Store the regression coefficient as $t_{n,a}$

- 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:

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



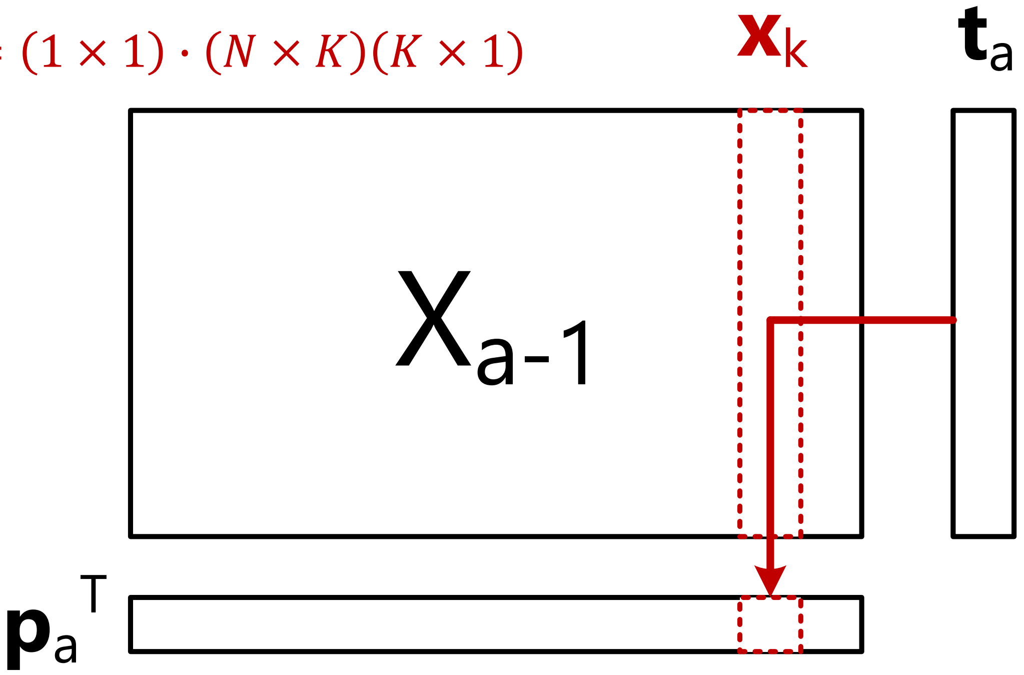
NIPALS Step By Step

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

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

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

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



NIPALS Step By Step

- CHECK FOR CONVERGENCE
- WORKSHOP – What are some methods we can use?
  - Compare $\boldsymbol{t}_a$ to $\boldsymbol{t}_a$ from previous iteration
  - Stop if $\Delta \boldsymbol{t}_a \leq \sqrt{\epsilon} \approx 10^{-8}$ (**demo in MATLAB**)
  - Change could mean $\left\|\boldsymbol{t}_a^i - \boldsymbol{t}_a^{i-1}\right\| \leq \sqrt{\epsilon}$ [absolute]
  - Change could mean $\frac{\left\|\boldsymbol{t}_a^i - \boldsymbol{t}_a^{i-1}\right\|}{\left\|\boldsymbol{t}_a^{i-1}\right\|} \leq \sqrt{\epsilon}$ [relative]
  - ALSO probably want an iterations limit in case (500 is good)

- AT CONVERGENCE
  - t_a and p_a are the a^{th} component
  - Store in T and P, respectively!

NIPALS Step By Step

- STEP 3 Deflate X_{a-1} to achieve X_a
- Deflation means *removing the part we can explain*
  - What can we explain? Why, $\hat{X}_a$ of course!
  - $E_a = X_{a-1} - \hat{X}_a \quad \Rightarrow \qquad E_a = X_{a-1} - \boldsymbol{t}_a \boldsymbol{p}_a^T$
  - E_a are the residuals after fitting the a^{th} component
  - Therefore, let $X_a = E_a$ and repeat from step 1 for $a + 1$
  - Example: for $a = 1$, use X_0 which is preprocessed data
  - Example: for $a = 2$, use X_1 which are residuals after 1 component

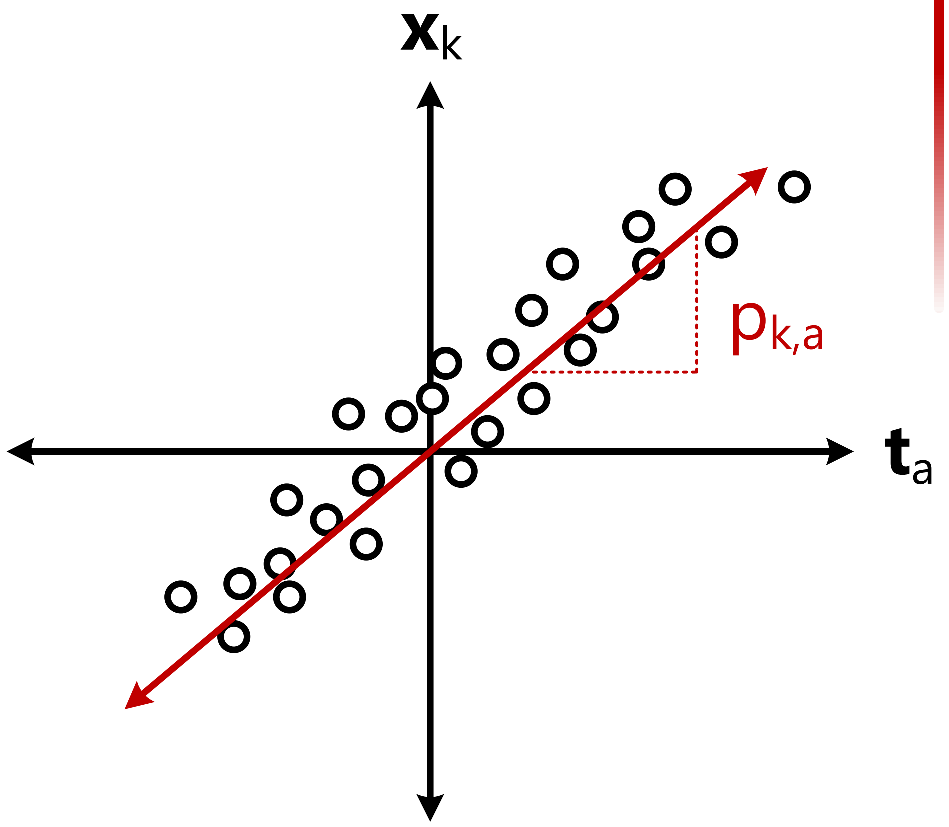
- A discussion about orthogonality
  - How do we know that subsequent components are orthogonal to those that came before it?

Interpretation of Regression Steps

- Let's teleport back to STEP 2.1 for a minute
  - STEP 2.1 Regress each column of X_{a-1} ($\boldsymbol{x}_k$) onto $\boldsymbol{t}_a$

- $p_{k,a} = \dfrac{\boldsymbol{t}_a^T \boldsymbol{x}_k}{\boldsymbol{t}_a^T \boldsymbol{t}_a}$
  - What does regression look like for a strong relationship?
  - Weak relationship?
  - What does that mean about $p_{k,a}$?
  - Regression can be used to predict $\hat{\boldsymbol{x}} = \boldsymbol{t}_a^T p_{k,a}$

- WORKSHOP: interpret step 2.3 as a linear regression
  - STEP 2.3 Repeat regression for all rows in X_{a-1}



Properties After Convergence

- Dropping subscripts (for simplicity), we have
  - $\boldsymbol{p} = \frac{\boldsymbol{t}^T X}{\boldsymbol{t}^T \boldsymbol{t}}$ $\qquad$ $\boldsymbol{t} = \frac{X\boldsymbol{p}}{\boldsymbol{p}^T\boldsymbol{p}}$
  - Recall that $\boldsymbol{p}$ is of unit length, thus $\boldsymbol{p}^T\boldsymbol{p} = 1$
  - Substitute $\boldsymbol{t}$ into equation for $\boldsymbol{p}$ to get:
  - $\boldsymbol{p} = \frac{X^T X\boldsymbol{p}}{\boldsymbol{t}^T t}$ $\qquad$ *OR* $\qquad$ $\boldsymbol{t}^T\boldsymbol{t}\,\boldsymbol{p} = X^T X\boldsymbol{p}$
  - This gives $(X^T X - \boldsymbol{t}^T\,\boldsymbol{t}\,I_K)\boldsymbol{p} = 0$ where I_K is a $K \times K$ identity

- And therefore[*]…
  - $\boldsymbol{p}$ is an eigenvector of $X^T X$
  - The eigenvalue for that eigenvector is $\lambda = \boldsymbol{t}^T\boldsymbol{t}$, which we know to be the variance explained in $\boldsymbol{t}$

**A WIIIITCH!*

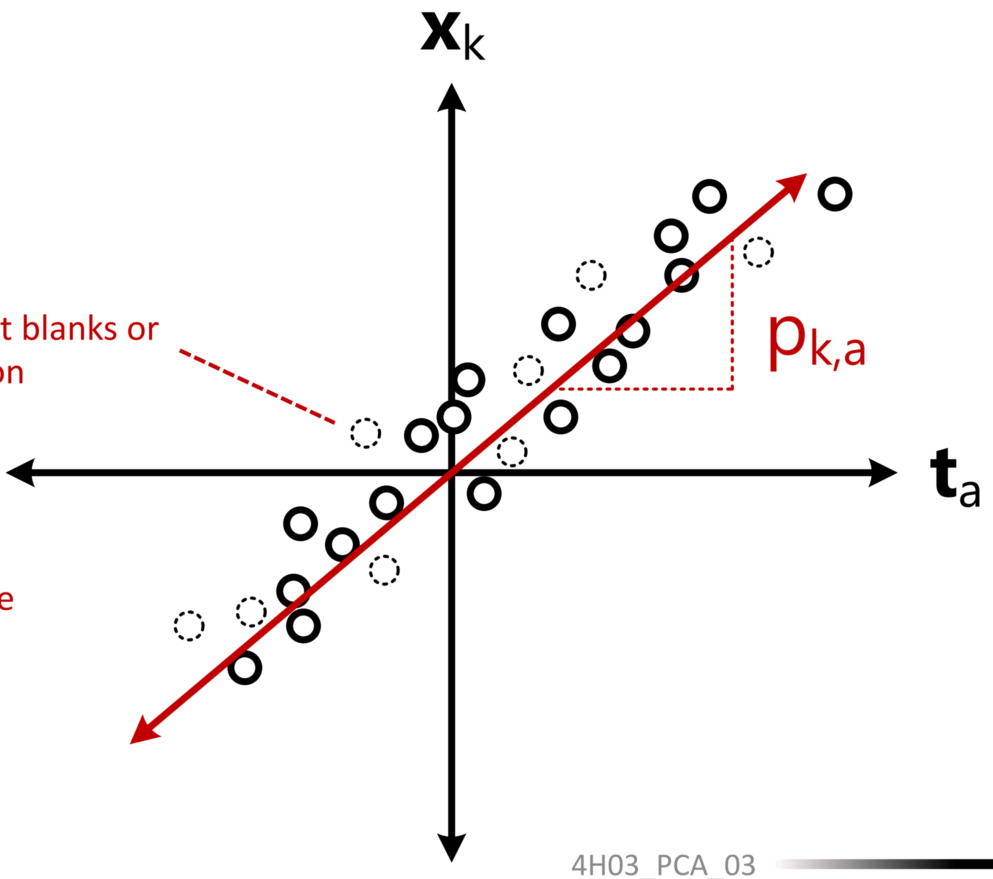
General NIPALS Comments

- In general, we can make the following remarks:
  - Convergence of NIPALS is **guaranteed** if left long enough
  - Convergence is fast if eigenvalues (variance explained by $\boldsymbol{t}$) are well separated
    - In other words, each component fits a new source of variance
  - If two eigenvalues are close (nearly equal variance explained by each component), convergence will be slow for the first component and fast for the second
  - NIPALS is capable of handling missing data
    - More on this now…

Handling Missing Data

- Missing values are simply **ignored**
  - **WORKSHOP**: What does this mean in our algorithm?
  - **WORKSHOP**: How would you implement this in practice?

These guys are actually just blanks or "`NaN`" in MATLAB or Python



QUESTION – what would happen if we tried to use EV decomposition with missing data?

Final Remarks

- NIPALS: The GOOD
  - Calculates ONE component at a time
  - Can handle missing data
  - Convergence guaranteed
- NIPALS: The BAD
  - Round-off errors will **accumulate**
  - Can suffer from outliers
- Other points to note
  - $\hat{X} = \boldsymbol{t}\,\boldsymbol{p}^T \equiv (-\boldsymbol{t})(-\boldsymbol{p}^T)$
  - Flipping signs does not matter; result will be the same
  - Can happen due to initial guesses, computer, blah blah

Next Up…

- Using model fitting statistics to improve accuracy
  - Filtering outliers from dataset
  - Using PCA on-line for soft sensors

- And finally, extension of PCA to PLS
  - Not much more, just one extra regression step in NIPALS!