# CPSC 340:
# Machine Learning and Data Mining

Principal Component Analysis

Fall 2019

# Last Time: MAP Estimation

- MAP estimation maximizes posterior:

$$\underbrace{p(w \mid X, y)}_{\text{"posterior"}} \propto \underbrace{p(y \mid X, w)}_{\text{"likelihood"}} \underbrace{p(w)}_{\text{"prior"}}$$

- Likelihood measures probability of labels ‘y’ given parameters ‘w’.
- Prior measures probability of parameters ‘w’ before we see data.
- For IID training data and independent priors, equivalent to using:

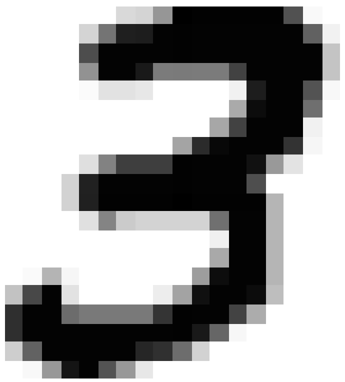
$$f(w) = -\sum_{i=1}^{n} \log(p(y_i \mid x_i, w)) - \sum_{j=1}^{d} \log(p(w_j))$$

- So log-likelihood is an error function, and log-prior is a regularizer.
  - Squared error comes from Gaussian likelihood.
  - L2-regularization comes from Gaussian prior.

# Motivation: Human vs. Machine Perception

- Huge difference between what we see and what computer sees:

What we see:

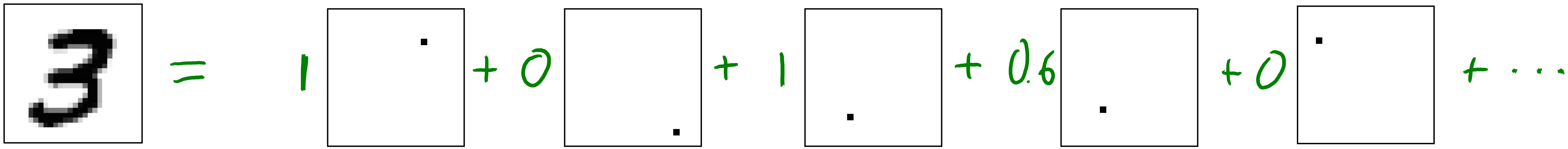


What the computer "sees":

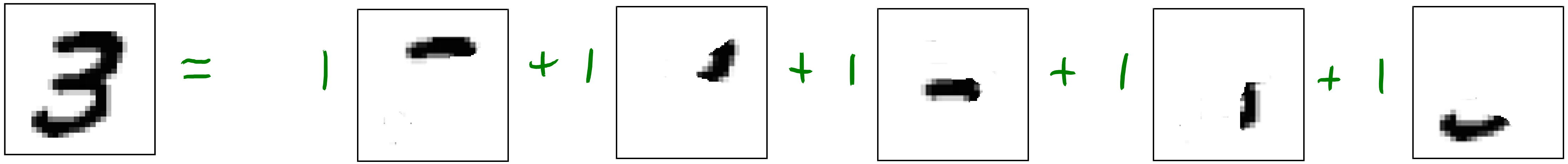
- But maybe images shouldn't be written as combinations of pixels.
  - Can we learn a better representation?
  - In other words, can we learn good features?

# Motivation: Pixels vs. Parts

- Can view 28x28 image as weighted sum of “single pixel on” images:



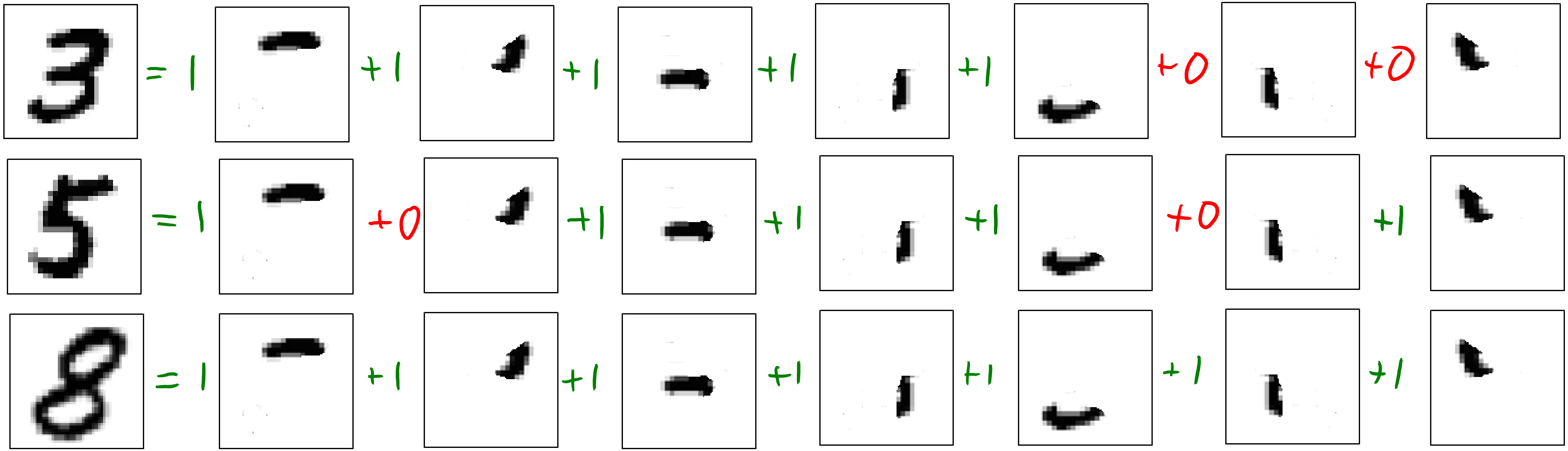

  - We have one image/feature for each pixel.
  - The weights specify “how much of this pixel is in the image”.
    - A weight of zero means that pixel is white, a weight of 1 means it’s black.
- This is non-intuitive, isn’t a “3” made of small number of “parts”?




  - Now the weights are “how much of this part is in the image”.

# Motivation: Pixels vs. Parts

- We could represent other digits as different combinations of "parts":




- Consider replacing images $x_i$ by the weights $z_i$ of the different parts:
  - The 784-dimensional $x_i$ for the "5" image is replaced by 7 numbers: $z_i$ = [1 0 1 1 1 0 1].
  - Features like this could make learning much easier.

# Part 4: Latent-Factor Models

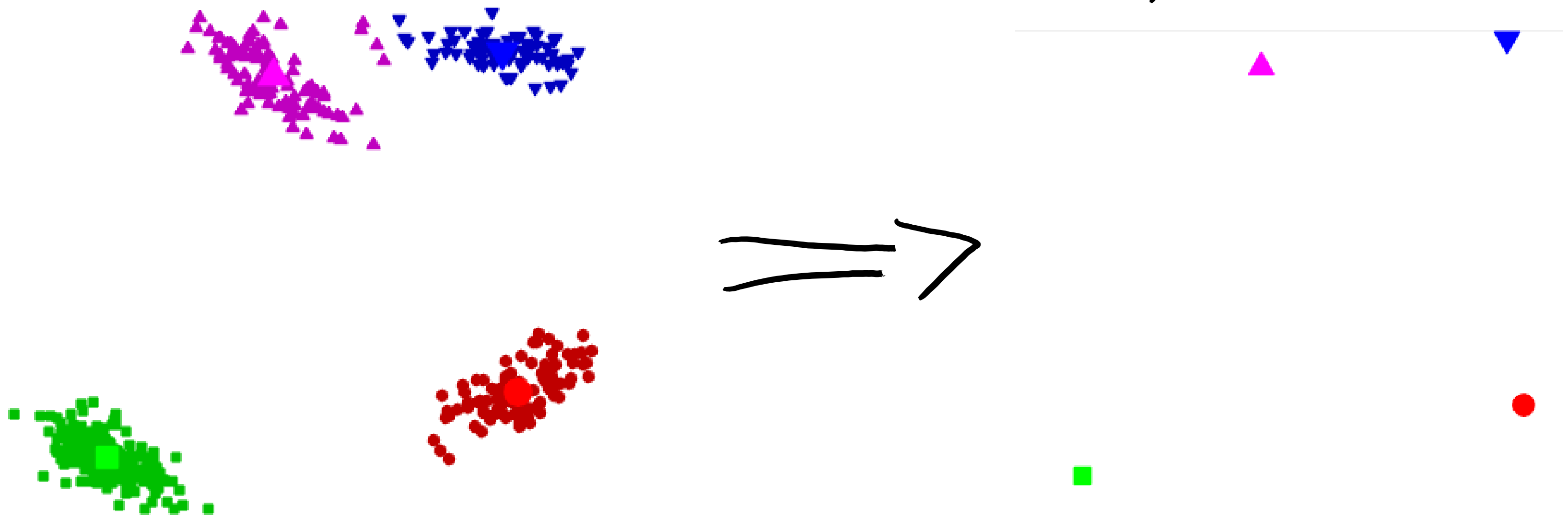
- The "part weights" are a change of basis from $x_i$ to some $z_i$.
  - But in high dimensions, it can be hard to find a good basis.
- Part 4 is about learning the basis from the data.

$$X \rightarrow \boxed{\text{Latent-factor model}} \rightarrow Z$$

- Why?
  - Supervised learning: we could use "part weights" as our features.
  - Outlier detection: it might be an outlier if isn't a combination of usual parts.
  - Dimension reduction: compress data into limited number of "part weights".
  - Visualization: if we have only 2 "part weights", we can view data as a scatterplot.
  - Interpretation: we can try and figure out what the "parts" represent.

# Previously: Vector Quantization

- Recall using k-means for vector quantization:
  - Run k-means to find a set of "means" $w_c$.
  - This gives a cluster $\hat{y}_i$ for each object 'i'.
  - Replace features $x_i$ by mean of cluster: $\hat{x}_i \approx w_{\hat{y}_i}$



- This can be viewed as a (really bad) latent-factor model.

# Vector Quantization (VQ) as Latent-Factor Model

- When d=3, we could write $x_i$ exactly as:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \end{bmatrix} = z_{i1}\underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{\text{"part 1"}} + z_{i2}\underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{\text{"part 2"}} + z_{i3}\underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\text{"part 3"}}$$

(this is like "one pixel on" representation of images)

  – In this "pointless" latent-factor model we have $z_i = [x_{i1}\ x_{i2}\ x_{i3}]$.

- If $x_i$ is in cluster 2, VQ approximates $x_i$ by mean $w_2$ of cluster 2:

$$x_i \approx w_2 = 0w_1 + 1w_2 + 0w_3 + 0w_4$$

- So in this example we would have $z_i = [0\ 1\ 0\ 0]$.
  – The "parts" are the means from k-means.
  – VQ only uses one part (the "part" from the cluster).

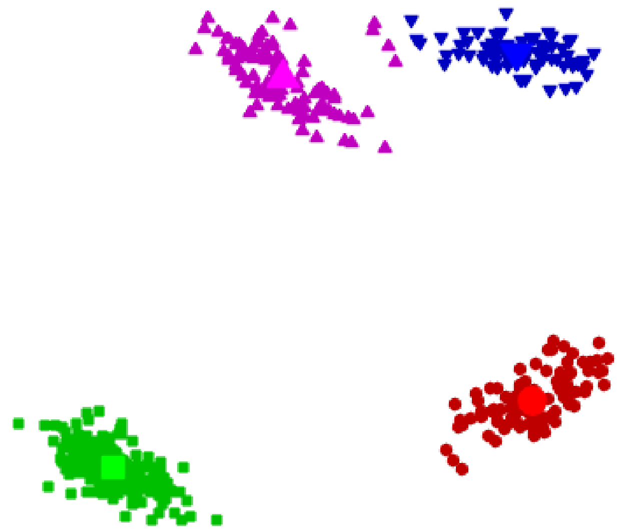
# Vector Quantization vs. PCA

- Viewing vector quantization as a latent-factor model:

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix} \longrightarrow \boxed{\text{vector quantization}} \longrightarrow Z = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ & \vdots & & \end{bmatrix}$$

"Parts" $w_1, w_2, \dots, w_K$

- Suppose we're doing supervised learning, and the colours are the true labels 'y':
  - Classification would be really easy with this "k-means basis" 'Z'.



# Vector Quantization vs. PCA

- Viewing vector quantization as a latent-factor model:

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix} \longrightarrow \boxed{\text{vector quantization}} \longrightarrow Z = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ & \vdots & & \end{bmatrix}$$

"Parts" $w_1, w_2, \dots, w_k$

- But it only uses 1 part, it's just memorizing 'k' points in $x_i$ space.
  – What we want is combinations of parts.
- PCA is a generalization that allows continuous '$z_i$':
  – It can have more than 1 non-zero.
  – It can use fractional weights and negative weights.

$$Z = \begin{bmatrix} 0.2 & 1.6 \\ 0.3 & 1.5 \\ 0.1 & -2.7 \\ 0.3 & -2.7 \\ \vdots & \end{bmatrix}$$

# Principal Component Analysis (PCA) Applications

- Principal component analysis (PCA) has been invented many times:

PCA was invented in 1901 by Karl Pearson,[1] as an analogue of the principal axis theorem in mechanics; it was later independently developed (and named) by Harold Hotelling in the 1930s.[2] Depending on the field of application, it is also named the discrete Kosambi-Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of $\mathbf{X}$ (Golub and Van Loan, 1983), eigenvalue decomposition (EVD) of $\mathbf{X}^T\mathbf{X}$ in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of [3]), Eckart–Young theorem (Harman, 1960), or Schmidt–Mirsky theorem in psychometrics, empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition (Sirovich, 1987), empirical component analysis (Lorenz, 1956), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

standard deviation of 3 in roughly the (0.878, 0.478) direction and of 1 in th orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the squa root of the corresponding eigenvalue, and shifted so their tails are at the mean.

https://en.wikipedia.org/wiki/Principal_component_analysis

# PCA Notation (MEMORIZE)

- PCA takes in a matrix 'X' and an input 'k', and outputs two matrices:

$$Z = \underbrace{\begin{bmatrix} - & z_1^T & - \\ - & z_2^T & - \\ & \vdots & \\ - & z_n^T & - \end{bmatrix}}_{k} \left.\vphantom{\begin{bmatrix}a\\a\\a\\a\end{bmatrix}}\right\} n \qquad W = \underbrace{\begin{bmatrix} - & w_1^T & - \\ - & w_2^T & - \\ & \vdots & \\ - & w_k^T & - \end{bmatrix}}_{d} \left.\vphantom{\begin{bmatrix}a\\a\\a\\a\end{bmatrix}}\right\} k = \underbrace{\begin{bmatrix} | & | & & | \\ w^1 & w^2 & \cdots & w^d \\ | & | & & | \end{bmatrix}}_{d} \left.\vphantom{\begin{bmatrix}a\\a\\a\end{bmatrix}}\right\} k$$

- For row 'c' of W, we use the notation $w_c$.
  - Each $w_c$ is a "part" (also called a "factor" or "principal component").
- For row 'i' of Z, we use the notation $z_i$.
  - Each $z_i$ is a set of "part weights" (or "factor loadings" or "features").
- For column 'j' of W, we use the notation $w^j$.
  - Index 'j' of all the 'k' "parts" (value of pixel 'j' in all the different parts).

# PCA Notation (MEMORIZE)

- PCA takes in a matrix 'X' and an input 'k', and outputs two matrices:

$$Z = \underbrace{\begin{bmatrix} — z_1^T — \\ — z_2^T — \\ \vdots \\ — z_n^T — \end{bmatrix}}_{k} \Big\} n \qquad W = \underbrace{\begin{bmatrix} — w_1^T — \\ — w_2^T — \\ \vdots \\ — w_k^T — \end{bmatrix}}_{d} \Big\} k = \underbrace{\begin{bmatrix} | & | & & | \\ w^1 & w^2 & \cdots & w^d \\ | & | & & | \end{bmatrix}}_{d} \Big\} k$$

- With this notation, we can write our approximation of one $x_{ij}$ as:

$$\hat{x}_{ij} = z_{i1} w_{1j} + z_{i2} w_{2j} + \cdots + z_{ik} w_{kj} = \sum_{c=1}^{k} z_{ic} w_{cj} = (w^j)^T z_i = \langle w^j, z_i \rangle$$

(NEW NOTATION)

  - K-means: "take index 'j' of closest mean".
  - PCA: "$z_i$ gives weights for index 'j' of all means".

- We can write approximation of the vector $x_i$ as:

$$\underset{d \times 1}{\hat{x}_i} = \begin{bmatrix} \langle w^1, z_i \rangle \\ \langle w^2, z_i \rangle \\ \vdots \\ \langle w^d, z_i \rangle \end{bmatrix} = \underset{d \times k}{W^T} \underset{k \times 1}{z_i}$$

# Different views (MEMORIZE)

- PCA approximates each $x_{ij}$ by the inner product $< w^j, z_i >$.
- PCA approximates each $x_i$ by the matrix-vector product $W^T z_i$.
- PCA approximates matrix 'X' by the matrix-matrix product ZW.

$$\underset{n \times d}{X} \approx \underset{n \times k}{Z}\,\underset{k \times d}{W}$$

  – PCA is also called a "matrix factorization" model.
  – Both 'Z' and 'W' are variables.

- This can be viewed as a "change of basis" from $x_i$ to $z_i$ values.
  – The "basis vectors" are the rows of W, the $w_c$.
  – The "coordinates" in the new basis of each $x_i$ are the $z_i$.

# PCA Applications

- Applications of PCA:
  - Dimensionality reduction: replace ‘X’ with lower-dimensional ‘Z’.
    - If k << d, then compresses data.
    - Often better approximation than vector quantization.

$$n\left\{\underbrace{\begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix}}_{d}\right. \approx n\left\{\underbrace{\begin{bmatrix} \\ Z \\ \\ \end{bmatrix}}_{k}\right. \underbrace{\begin{bmatrix} & W & \end{bmatrix}}_{d}\Big\}k$$

# PCA Applications

- Applications of PCA:
  - Dimensionality reduction: replace ‘X’ with lower-dimensional ‘Z’.
    - If k << d, then compresses data.
    - Often better approximation than vector quantization.

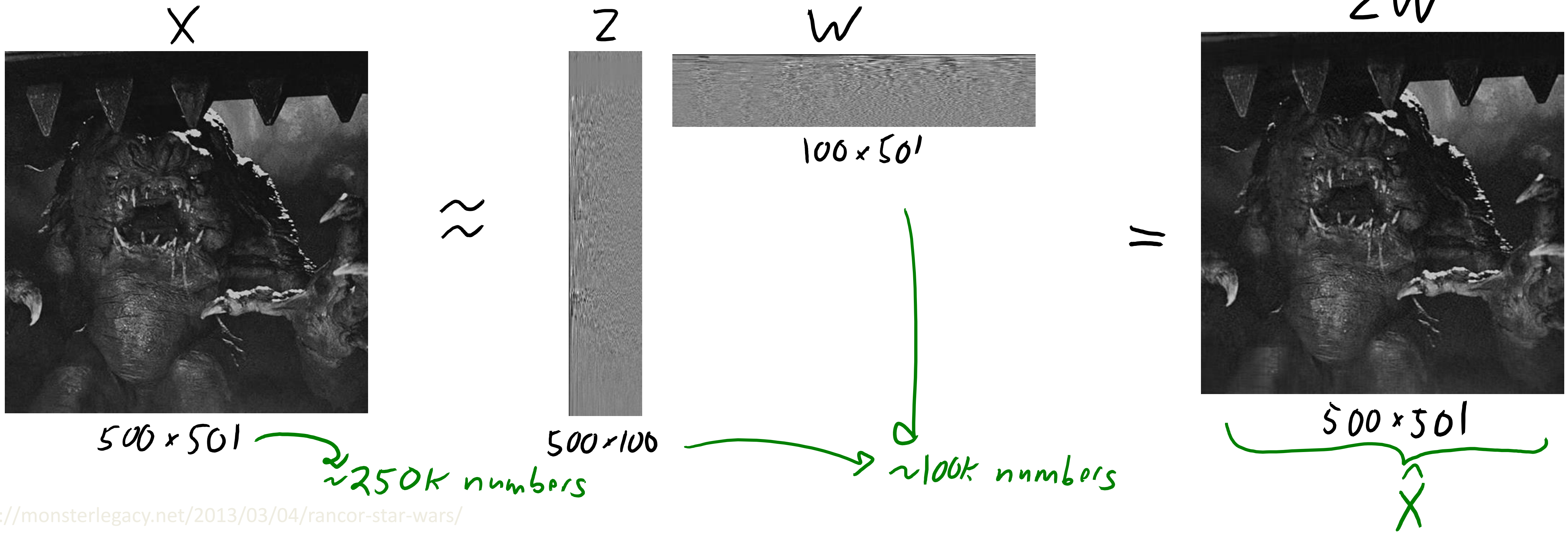
$$\overset{X}{\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 32 \\ 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 \\ 6 & 12 & 18 & 24 & 30 & 36 & 42 & 48 \\ 7 & 14 & 21 & 28 & 35 & 42 & 49 & 56 \\ 8 & 16 & 24 & 32 & 40 & 48 & 56 & 64 \end{bmatrix}} \approx \overset{Z}{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}} \overset{W}{\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}}$$

Compresses 64 elements of 'X' down to 16 elements of 'Z' and 'W'

(can predict all $x_i$ values from <u>one</u> $z_i$ value)

# PCA Applications

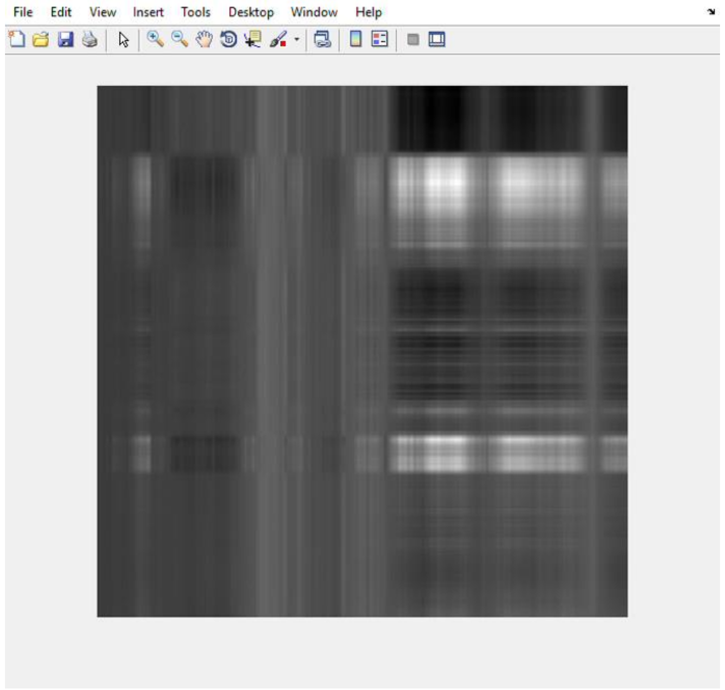
- Applications of PCA:
  - Dimensionality reduction: replace ‘X’ with lower-dimensional ‘Z’.
    - If k << d, then compresses data.
    - Often better approximation than vector quantization.




https://monsterlegacy.net/2013/03/04/rancor-star-wars/

# PCA Applications

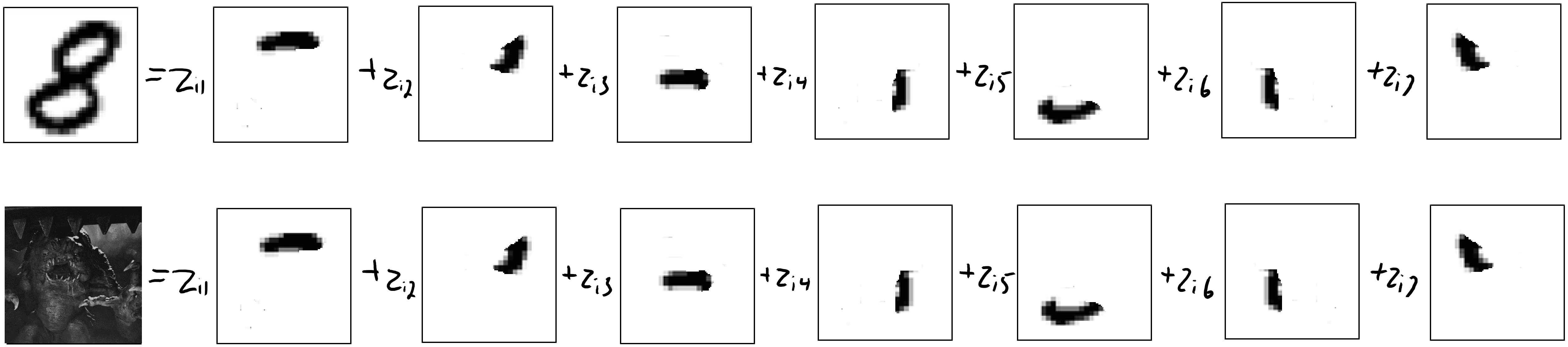
- Applications of PCA:
  - Dimensionality reduction: replace ‘X’ with lower-dimensional ‘Z’.
    - If k << d, then compresses data.
    - Often better approximation than vector quantization.




https://monsterlegacy.net/2013/03/04/rancor-star-wars/

# PCA Applications

- Applications of PCA:
  - Outlier detection: if PCA gives poor approximation of $x_i$, could be 'outlier'.
    - Though due to squared error PCA is sensitive to outliers.



# PCA Applications

- Applications of PCA:
  - Partial least squares: uses PCA features as basis for linear model.

Compute approximation $X \approx ZW$

Now use $Z$ as features in a linear model:

$$y_i = v^T z_i$$

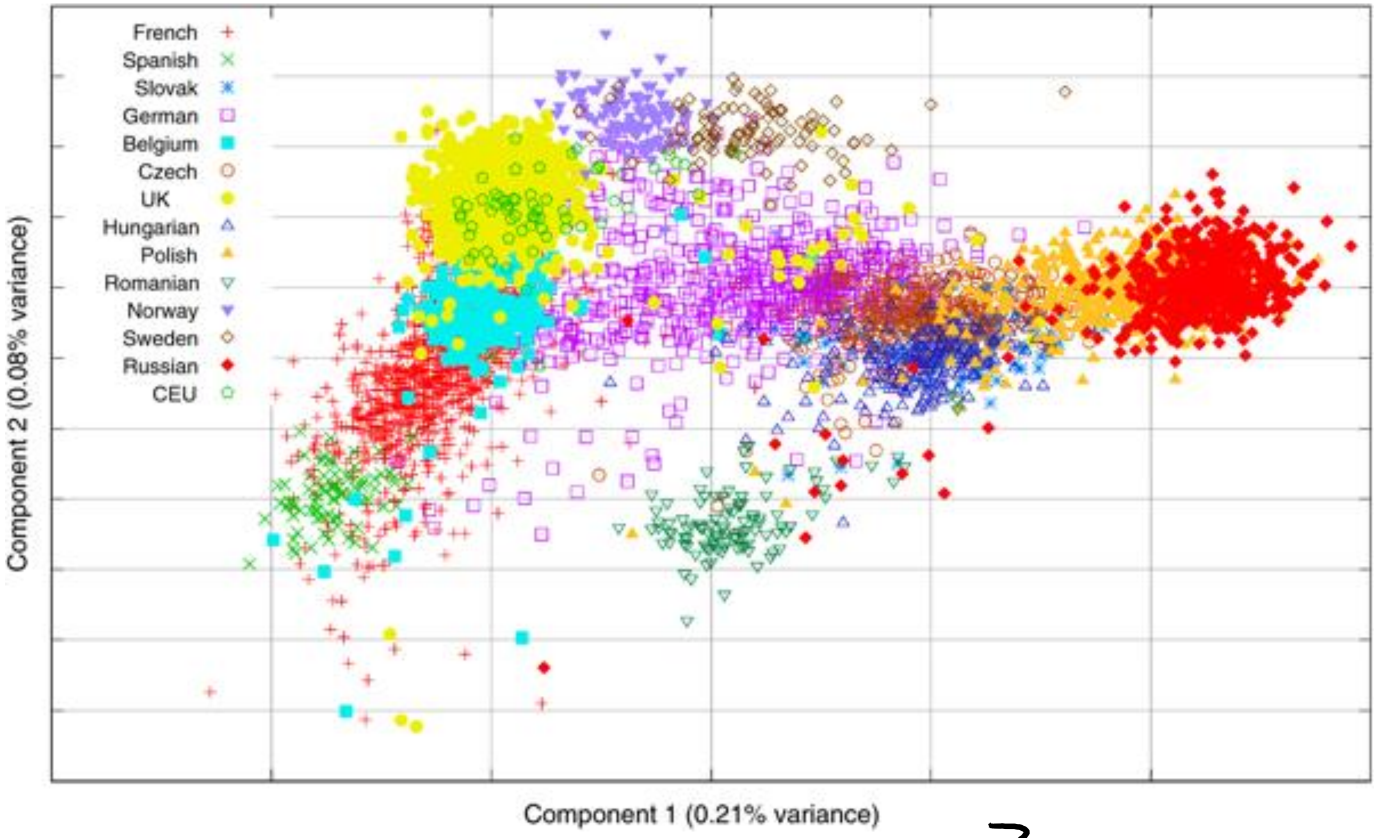
linear regression weights 'v' trained under this change of basis

lower-dimensional than original features so less overfitting

# PCA Applications

- Applications of PCA:
  - Data visualization: plot $z_i$ with $k = 2$ to visualize high-dimensional objects.

$z_{i2}$

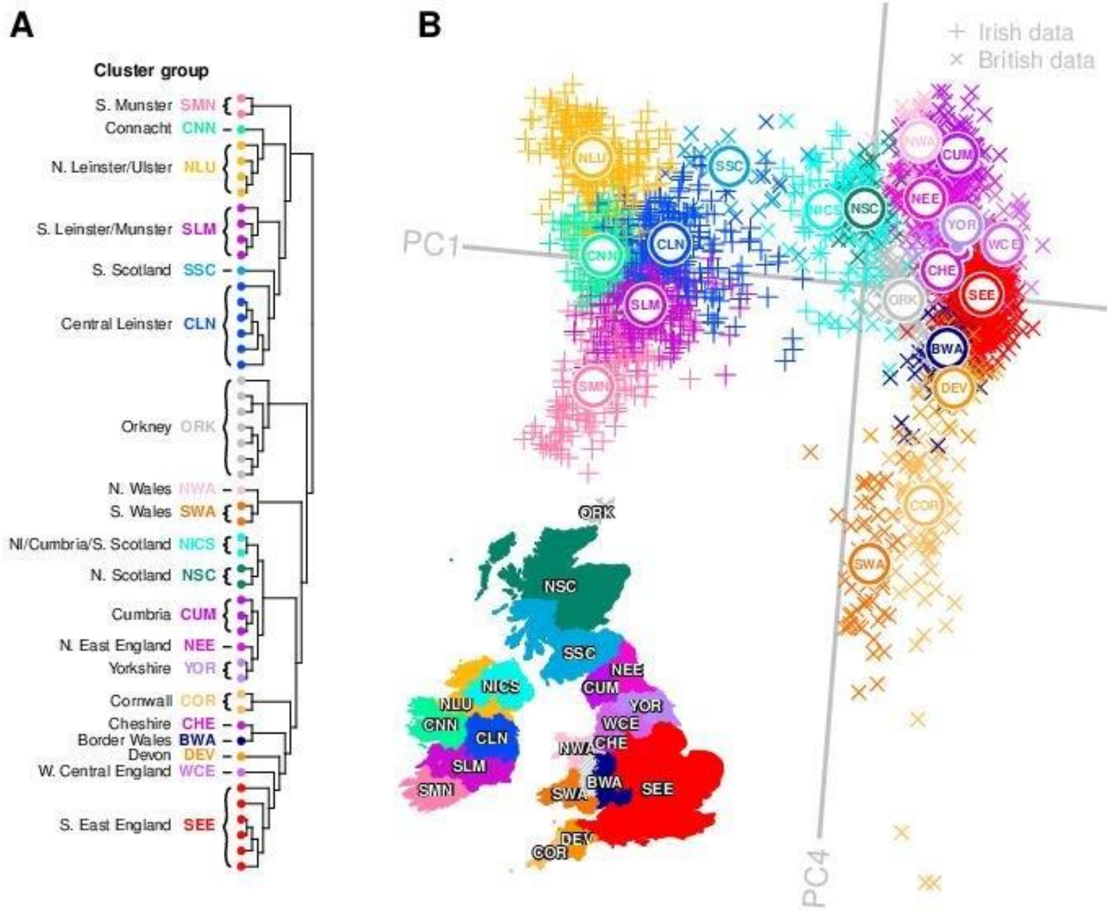



$z_{i1}$

http://infoproc.blogspot.ca/2008/11/european-genetic-substructure.html

# PCA Applications

- Applications of PCA:
  - Data visualization: plot $z_i$ with $k = 2$ to visualize high-dimensional objects.
    - Can augment other visualizations:




https://www.sciencedaily.com/releases/2018/01/180125140943.htm

# PCA Applications

- Applications of PCA:
  - Data interpretation: we can try to assign meaning to latent factors $w_c$.
    - Hidden "factors" that influence all the variables.

| Trait | Description |
|---|---|
| Openness | Being curious, original, intellectual, creative, and open to new ideas. |
| Conscientiousness | Being organized, systematic, punctual, achievement-oriented, and dependable. |
| Extraversion | Being outgoing, talkative, sociable, and enjoying social situations. |
| Agreeableness | Being affable, tolerant, sensitive, trusting, kind, and warm. |
| Neuroticism | Being anxious, irritable, temperamental, and moody. |

https://new.edu/resources/big-5-personality-traits

"Most Personality Quizzes Are Junk Science. I Found One That Isn't."

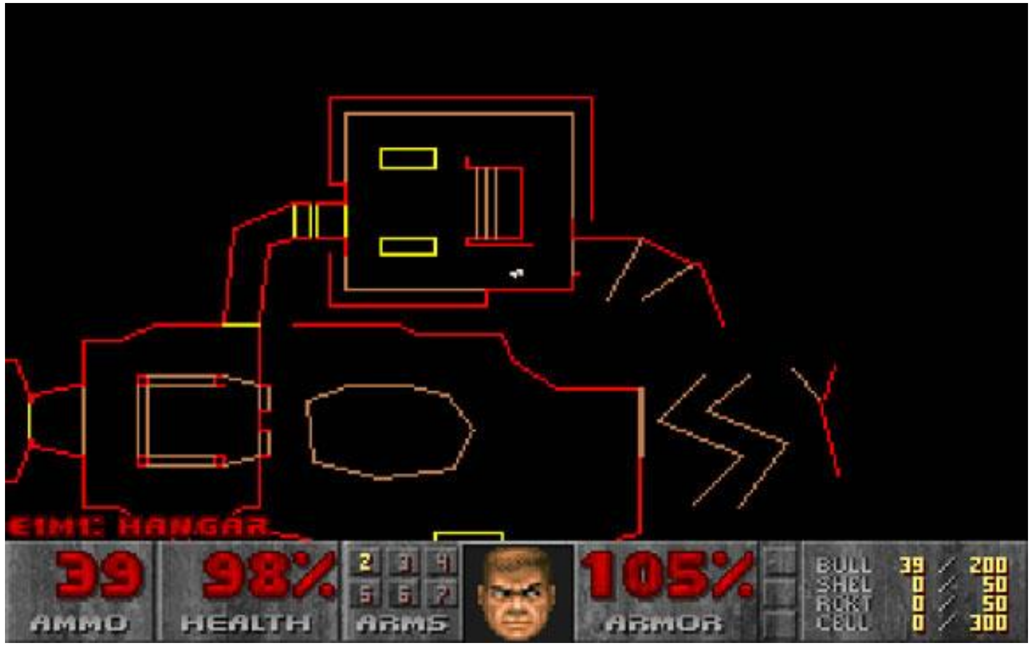
What is PCA actually doing?

When should PCA work well?

Today I just want to show geometry,
we’ll talk about implementation next time.

# Doom Overhead Map and Latent-Factor Models

- Original "Doom" video game included an "overhead map" feature:


- This map can be viewed as a latent-factor model of player location.

https://en.wikipedia.org/wiki/Doom_(1993_video_game)
https://forum.minetest.net/viewtopic.php?f=5&t=9666

# Overhead Map and Latent-Factor Models

- Actual player location at time 'i' can be described by 3 coordinates:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \end{bmatrix} \begin{array}{l} \leftarrow \text{"x" coordinate} \\ \leftarrow \text{"y" coordinate} \\ \leftarrow \text{"z" coordinate} \end{array}$$

- The overhead map approximates these 3 coordinates with only 2:

$$z_i = \begin{bmatrix} z_{i1} \\ z_{i2} \end{bmatrix} \begin{array}{l} \leftarrow \text{"x" coordinate} \\ \leftarrow \text{"y" coordinate} \end{array}$$

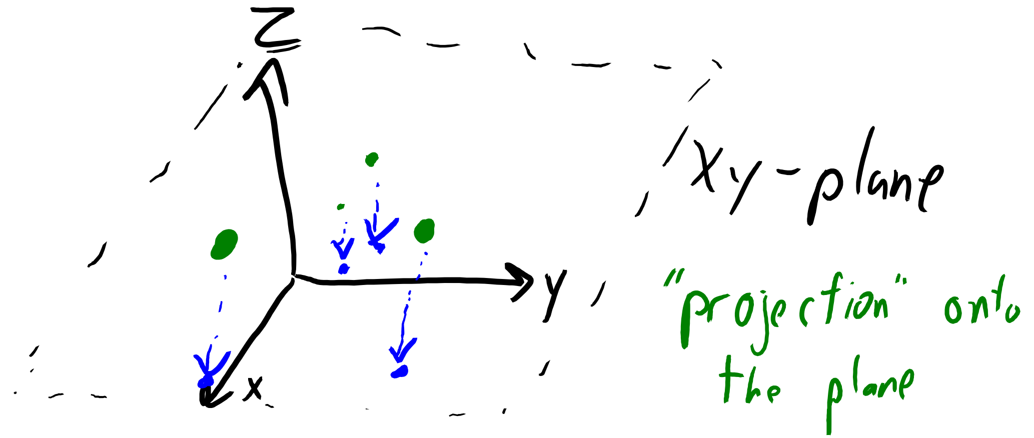
- Our k=2 latent factors are the following:

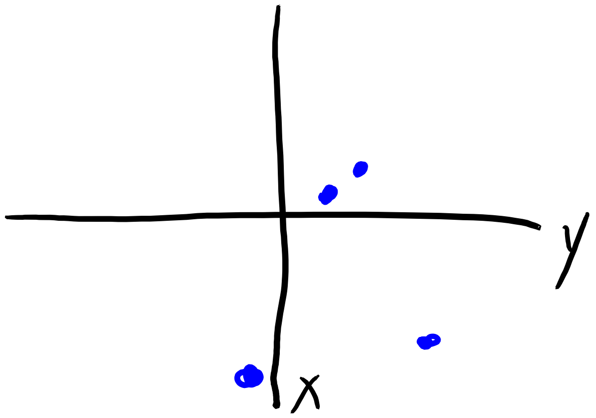
$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- So our approximation of $x_i$ is: $\hat{x}_i = z_{i1}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + z_{i2}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

# Overhead Map and Latent-Factor Models

- The "overhead map" approximation just ignores the "height".



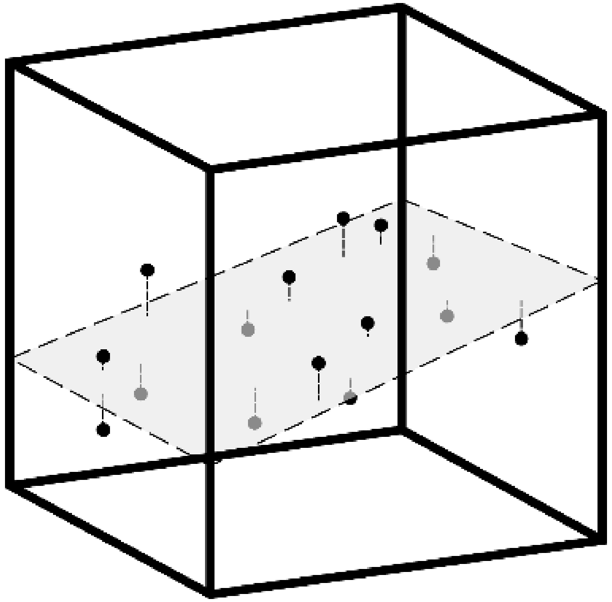




  - This is a good approximation if the world is flat.
    - Even if the character jumps, the first two features will approximate location.

  - But it's a poor approximation if heights are different.

# Overhead Map and Latent-Factor Models

- Consider these crazy goats trying to get some salt:
  – Ignoring height gives poor approximation of goat location.

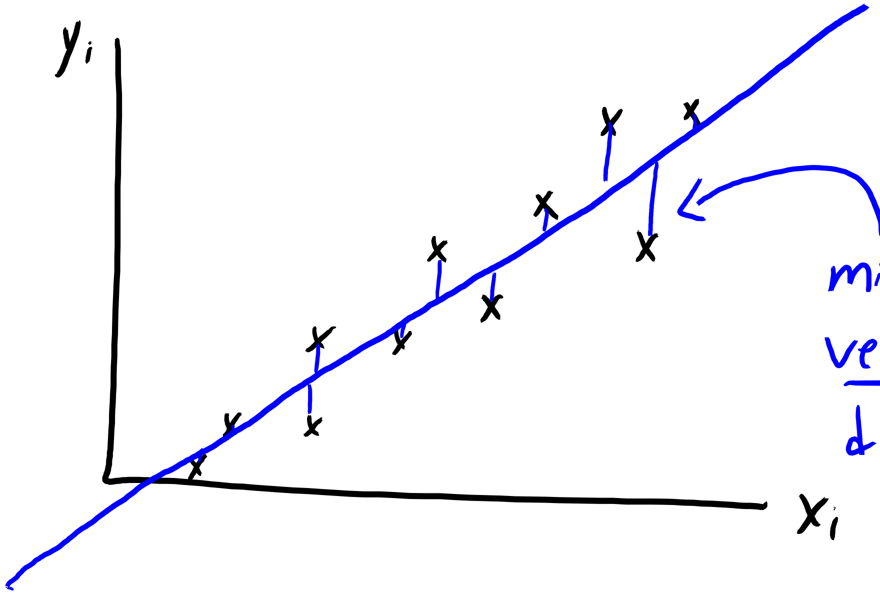


- But the "goat space" is basically a two-dimensional plane.
  – Better k=2 approximation: define 'W' so that combinations give the plane.

www.momtastic.com/webecoist/2010/11/07/some-fine-dam-climbing-goats-scaling-steep-vertical-wall
https://www.quora.com/What-is-a-simplified-explanation-and-proof-of-the-Johnson-Lindenstrauss-lemma

# PCA with d=2 and k =1

## Least squares

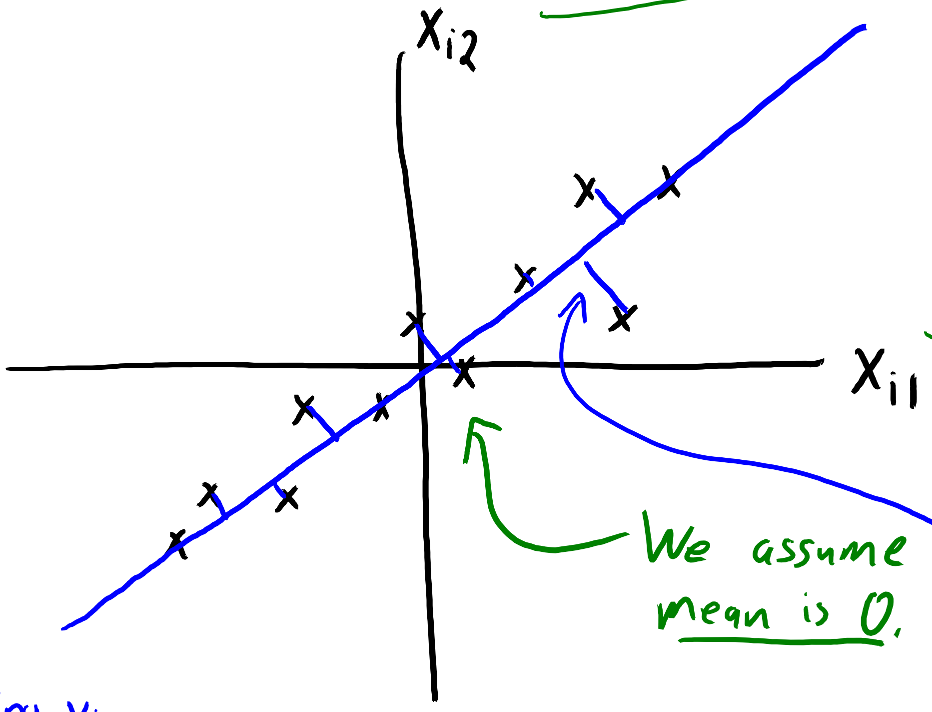



minimize vertical squared distance

We only care about predicting $y_i$

## Principal component analysis

We're predicting features.

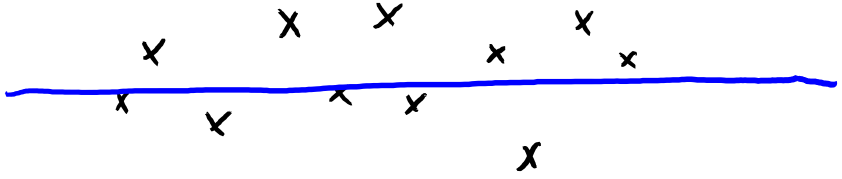



We assume mean is 0.

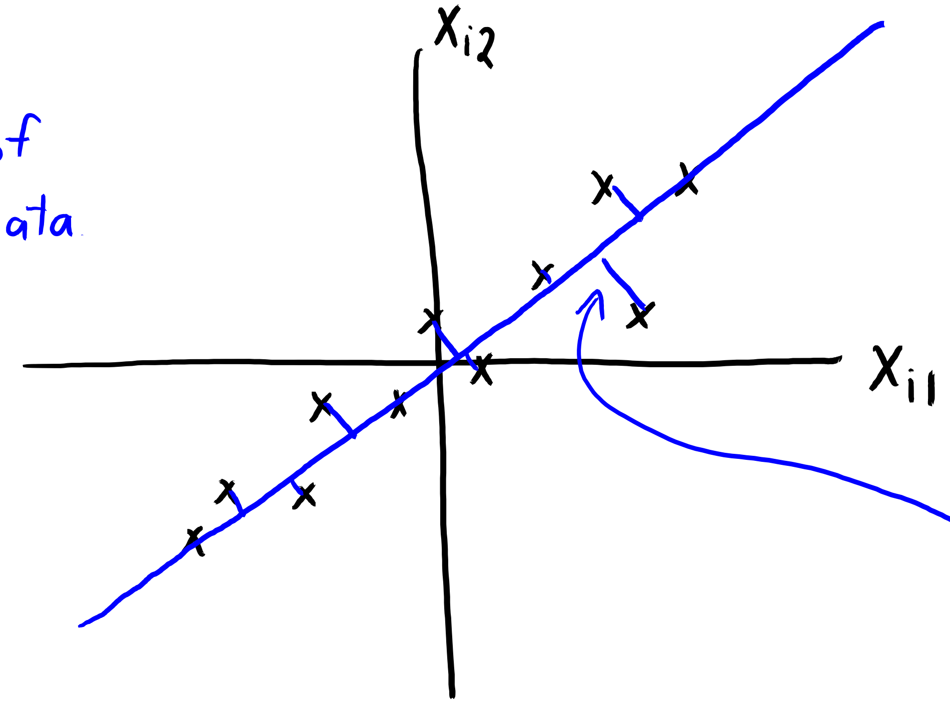
PCA finds line 'W' minimizing squared distance in both dimensions.

# PCA with d=2 and k =1

Principal component analysis



You can think of 'W' as <u>rotating</u> data.

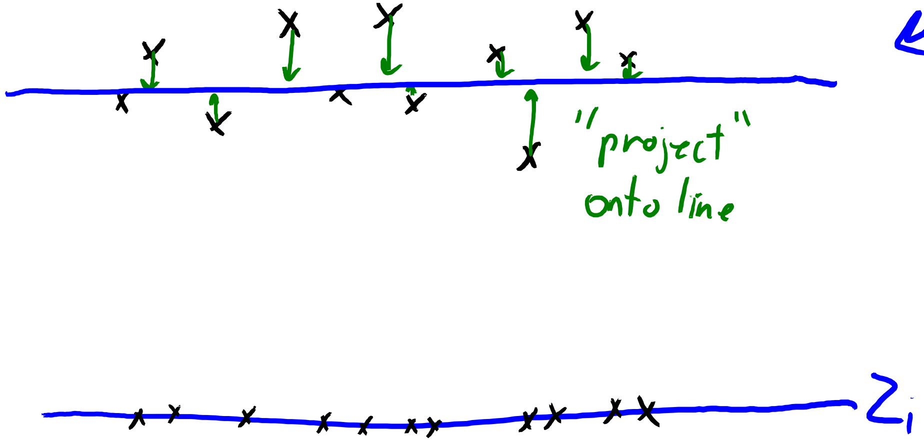


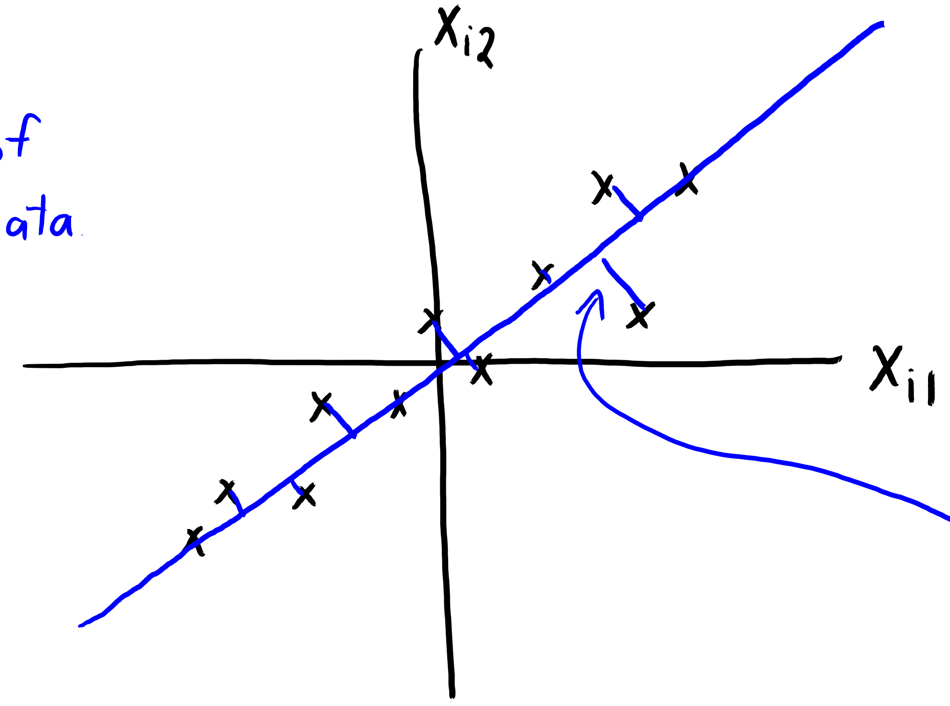

PCA finds line 'W' minimizing squared distance in <u>both</u> dimensions.

# PCA with d=2 and k =1

Principal component analysis

You can think of 'W' as rotating data.



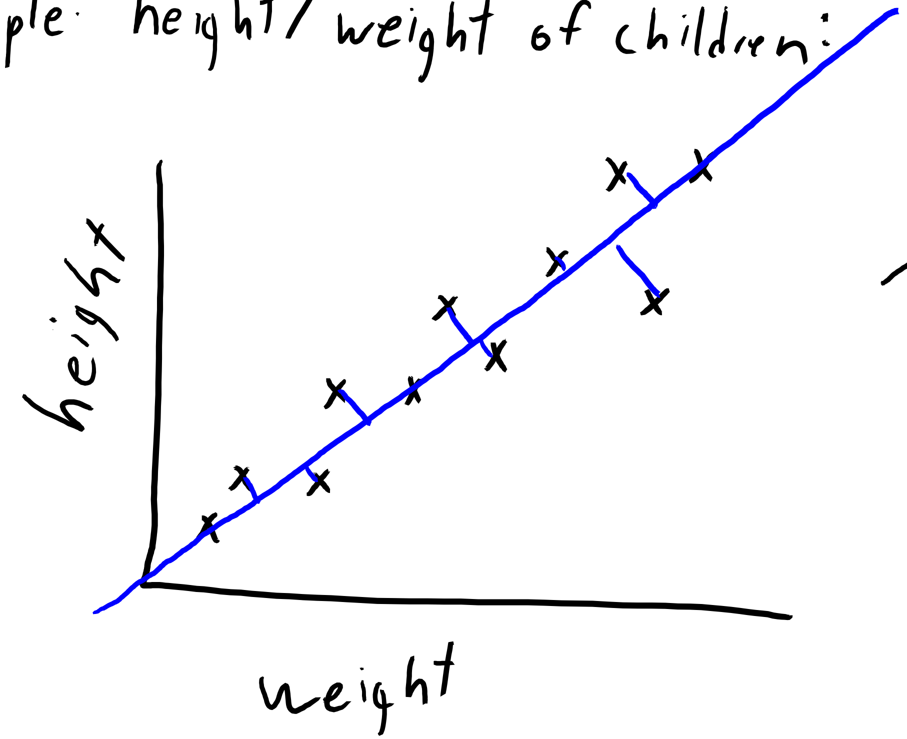

$z_i$ can be interpreted as position along the line.

(turned a 2d dataset into a 1d dataset)

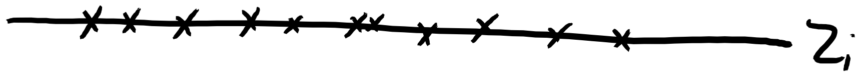
PCA finds line 'W' minimizing squared distance in both dimensions.

# PCA with d=2 and k =1

Example: height/weight of children:



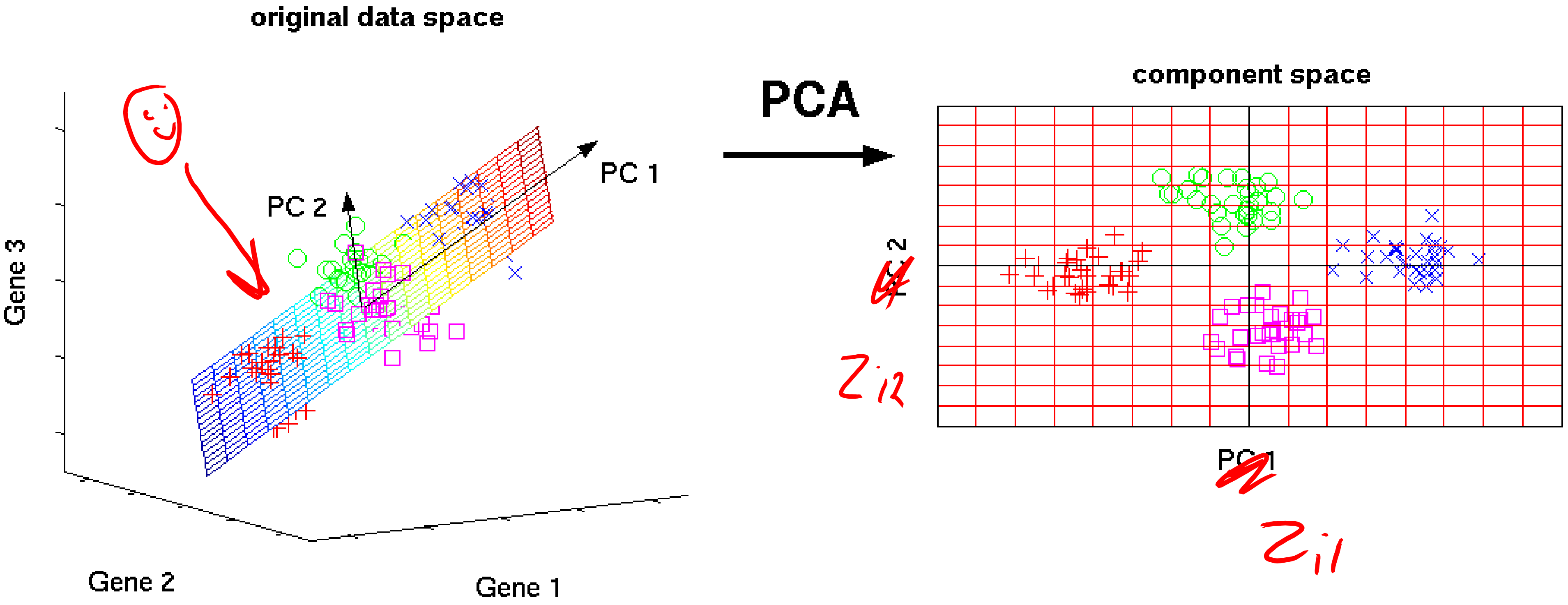

PCA with k=1




Latent factor could be viewed as measure of size.

# PCA with d=3 and k=2.

- With d=3, PCA (k=1) finds line minimizing squared distance to $x_i$.
- With d=3, PCA (k=2) finds plane minimizing squared distance to $x_i$.




http://www.nlpca.org/fig_pca_principal_component_analysis.png

# Summary

- Latent-factor models:
  - Try to learn basis Z from training examples X.
  - Usually, the $z_i$ are "part weights" for "parts" $w_c$.
  - Useful for dimensionality reduction, visualization, factor discovery, etc.
- Principal component analysis:
  - Writes each training examples as linear combination of parts.
    - We learn both the "parts" 'W' and the "features" Z.
  - We can view 'W' as best lower-dimensional hyper-plane.
  - We can view 'Z' as the coordinates in the lower-dimensional hyper-plane.

- Next time: PCA in 4 lines of code.