CPSC 340: Machine Learning and Data Mining

Principal Component Analysis

Fall 2016

Admin

- A2/Midterm:
  - Grades/solutions will be posted after class.
- Assignment 4:
  - Posted, due November 14.
- Extra office hours:
  - Thursdays from 4:30-5:30 in ICICS X836.

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 prior, 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.

Multi-Class Classification

- For binary classification with linear models we use:

$$y_i = \text{sign}(w^T x_i)$$

- For multi-class classification with linear models we use:

$$y_i = \underset{c}{\text{argmax}} \{ w_c^T x_i \}$$

  – Where we have a vector w_c for each class ‘c’.

$$W = \begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix}$$

- To jointly estimate the w_c, we can use softmax likelihood:

$$p(y_i = c \mid x_i, W) = \frac{\exp(w_c^T x_i)}{\sum_{c'=1}^{k} \exp(w_{c'}^T x_i)}$$

Multi-Class Classification

- For multi-class classification with linear models we use:

$$y_i = \underset{c}{\operatorname{argmax}} \{ w_c^T x_i \}$$

- To jointly estimate the w_c, we can use softmax likelihood:

$$p(y_i \mid x_i, W) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c=1}^{k} \exp(w_c^T x_i)}$$

- By taking the negative log and adding a regularizer, we get:

$$f(W) = \sum_{i=1}^{n} \Bigg[\underbrace{-w_{y_i}^T x_i}_{\text{Tries to make } w_c^T x_i \text{ big for the correct label}} + \underbrace{\log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right)}_{\text{Approximates } \max_c \{w_c^T x_i\} \text{ so tries to make } w_c^T x_i \text{ small for incorrect labels}} \Bigg] + \underbrace{\frac{\lambda}{2} \sum_{c=1}^{k} \sum_{j=1}^{d} w_{cj}^2}_{\text{Usual } L_2\text{-regularizer on elements of 'W'}}$$

Digression: Frobenius Matrix Norm

We can write $\sum_{i=1}^{n}\sum_{j=1}^{d} w_{ij}^2$ in matrix notation as $\|W\|_F^2$

The notation $\|W\|_F$ is the "Frobenius" norm of matrix W:

$$\|W\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{d} w_{ij}^2}$$

(L_2-norm if we "stack" columns of 'W' into a big vector)

End of Part 3: Key Concepts

- Linear models base predictions on linear combinations of features:

$$w^T x_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$

- We model non-linear effects using a change of basis:
  - Replace x_i with z_i and use $w^T z_i$.
  - Examples include polynomial basis and (non-parametric) RBFs.

- Regression is supervised learning with continuous labels.
  - Logical error measure for regression is squared error:

$$f(w) = \frac{1}{2}\|Xw - y\|^2$$

  - Can be solved as a system of linear equations.

End of Part 3: Key Concepts

- We can reduce over-fitting by using regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- Squared error is not always right measure:
  - Absolute error is less sensitive to outliers.
  - Logistic loss and hinge loss are better for binary y_i.
  - Softmax loss is better for multi-class y_i.
- MLE/MAP perspective:
  - We can view loss as log-likelihood and regularizer as log-prior.
  - Allows us to define losses based on probabilities.

End of Part 3: Key Concepts

- Gradient descent finds local minimum of smooth objectives.
  - Converges to a global optimum for convex functions.
  - Can use smooth approximations (Huber, log-sum-exp)
- Stochastic gradient methods allow huge/infinite ‘n’.
  - Though very sensitive to the step-size.
- Kernels let us use similarity between examples, instead of features.
  - Let us use some exponential- or infinite-dimensional features.
- Feature selection is a messy topic.
  - Classic methods are hypothesis testing and search and score.
  - L1-regularization simultaneously regularizes and selects features.

The Story So Far…

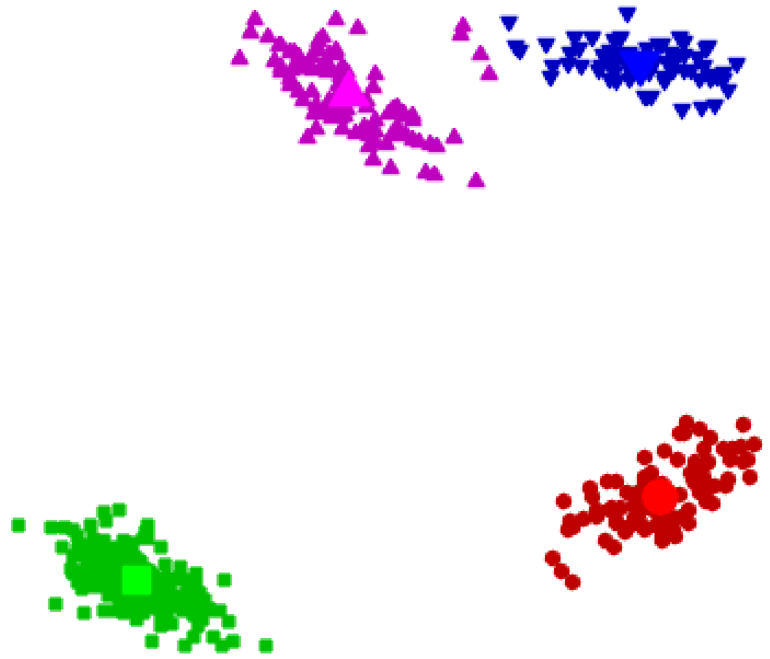
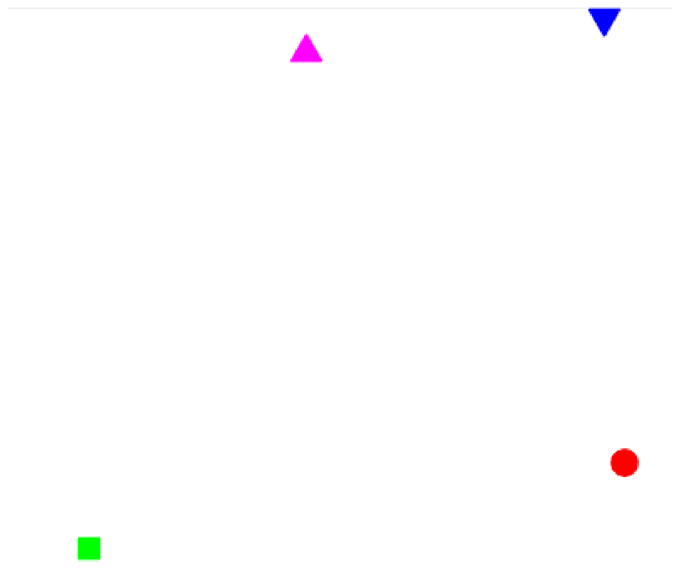
- Supervised Learning Part 1:
  - Methods based on counting and distances.
- Unsupervised Learning Part 1:
  - Methods based on counting and distances.
- Supervised Learning Part 2:
  - Methods based on linear models and gradient descent.
- Unsupervised Learning Part 2:
  - Methods based on linear models and gradient descent.

Unsupervised Learning Part 2

- Unsupervised learning:
  - We only have x_i values, but no explicit target labels.
  - You want to do 'something' with them.
- Some unsupervised learning tasks:
  - Clustering: What types of x_i are there?
  - Outlier detection: Is this a 'normal' x_i?
  - Association rules: Which x_{ij} occur together?
  - Latent-factors: What 'parts' are the x_i made from?
  - Data visualization: What does the high-dimensional X look like?
  - Ranking: Which are the most important x_i?

Motivation: Vector Quantization

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

Motivation: Vector Quantization

- We can write vector quantization as a linear model:
  - Define 'z_i' as a binary vector that is zero except in position c_i.

    If $k=4$ and $c_i=3$ then $z_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

  - Our weird notation for mean matrix 'W':

$$\underset{k \times d}{W} = \underbrace{\begin{bmatrix} \text{—} w_{c_1} \text{—} \\ \text{—} w_{c_2} \text{—} \\ \vdots \\ \text{—} w_{c_k} \text{—} \end{bmatrix}}_{\text{Each row is a mean.}} = \underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_d \\ | & | & & | \end{bmatrix}}_{\text{Each column is feature 'j' for each mean.}}$$

Weird notation alert:
- w_{c_i} is row c_i of W
- w_j is column c_i of W

So $w_{c_i} = \begin{bmatrix} w_1^T z_i \\ w_2^T z_i \\ \vdots \\ w_d^T z_i \end{bmatrix} = W^T z_i$ So vector quantization uses $x_{ij} \approx w_j^T z_i$ and $x_i \approx W^T z_i$

Regression View of K-Means

- Recall that we said k-means minimizes the objective:

$$f(W,c) = \sum_{i=1}^{n}\sum_{j=1}^{d}(w_{c_i j} - x_{ij})^2$$

- In our new notation, we can write k-means as minimizing:

$$f(W,Z) = \sum_{i=1}^{n}\sum_{j=1}^{d}(w_j^T z_i - x_{ij})^2 \qquad \text{where } Z = \overset{\text{Each row has } \underline{\text{1 non-zero}}}{\begin{bmatrix} — z_1^T — \\ — z_2^T — \\ \vdots \\ — z_n^T — \end{bmatrix}}$$

- We can view this as solving 'd' regression problems:
  - Each w_j is trying to predict column 'j' of 'X' from the basis z_i.
  - But we're also trying to learn the basis z_i.

Principal Component Analysis (PCA)

- Principal component analysis (PCA) minimizes the same objective:

$$f(W,Z) = \sum_{i=1}^{n} \sum_{j=1}^{d} (w_j^T z_i - x_{ij})^2$$

  - But instead of "1 of k" binary z_i we allow a continuous basis z_i.
- Called a latent-factor model:
  - Instead of means, $w_{c\,j}$ called "factors" or "principal components".
  - The z_i are called "factor loadings" or "low-dimensional basis".
    - The z_i say how to mix the means/factors to approximate example 'i'.

We don't just approximate x_{ij} by element 'j' of cluster mean w_{c_i}

We approximate x_{ij} ~~it~~ as linear combination of all means/factors/PCs at position 'j' : $x_{ij} \approx w_{j1}z_{i1} + w_{j2}z_{i2} + \dots + w_{jk}z_{ik}$

Principal Component Analysis (PCA)

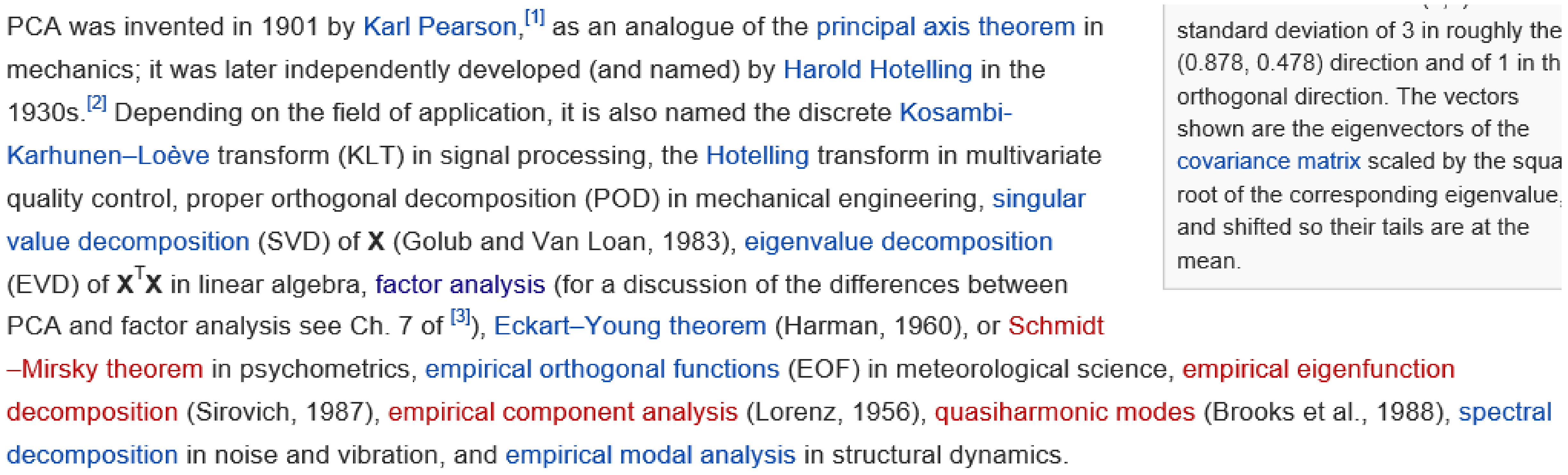
- Principal component analysis (PCA) in matrix notation:

$$
\begin{aligned}
f(W,Z) &= \sum_{i=1}^{n}\sum_{j=1}^{d}(w_j^T z_i - x_{ij})^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{d}(w_{j1}z_{i1} + w_{j2}z_{i2} + \dots + w_{jk}z_{ik} - x_{ij})^2 \\
&= \sum_{i=1}^{n}\|W^T z_i - x_i\|^2 \\
&= \|ZW - X\|_F^2
\end{aligned}
$$

- Also called a matrix factorization model: $\overset{n\times d}{X} \approx \overset{n\times k}{Z}\,\overset{k\times d}{W}$

PCA Applications

- PCA has been reinvented 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 **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 Applications

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

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

PCA Applications

- Applications of PCA:
  - Dimensionality reduction: replace 'X' with lower-dimensional 'Z'.
    - If k << d, then compresses data.
    - Much better approximation than vector quantization.
  - Outlier detection: if PCA gives poor approximation of x_i, could be 'outlier'.
    - Though due to squared error PCA is sensitive to outliers.
  - Partial least squares: uses PCA features as basis for linear model.

Compute approximation $X \approx ZW$

Now Z as features in a linear model:

$$y_i = w^T z_i$$

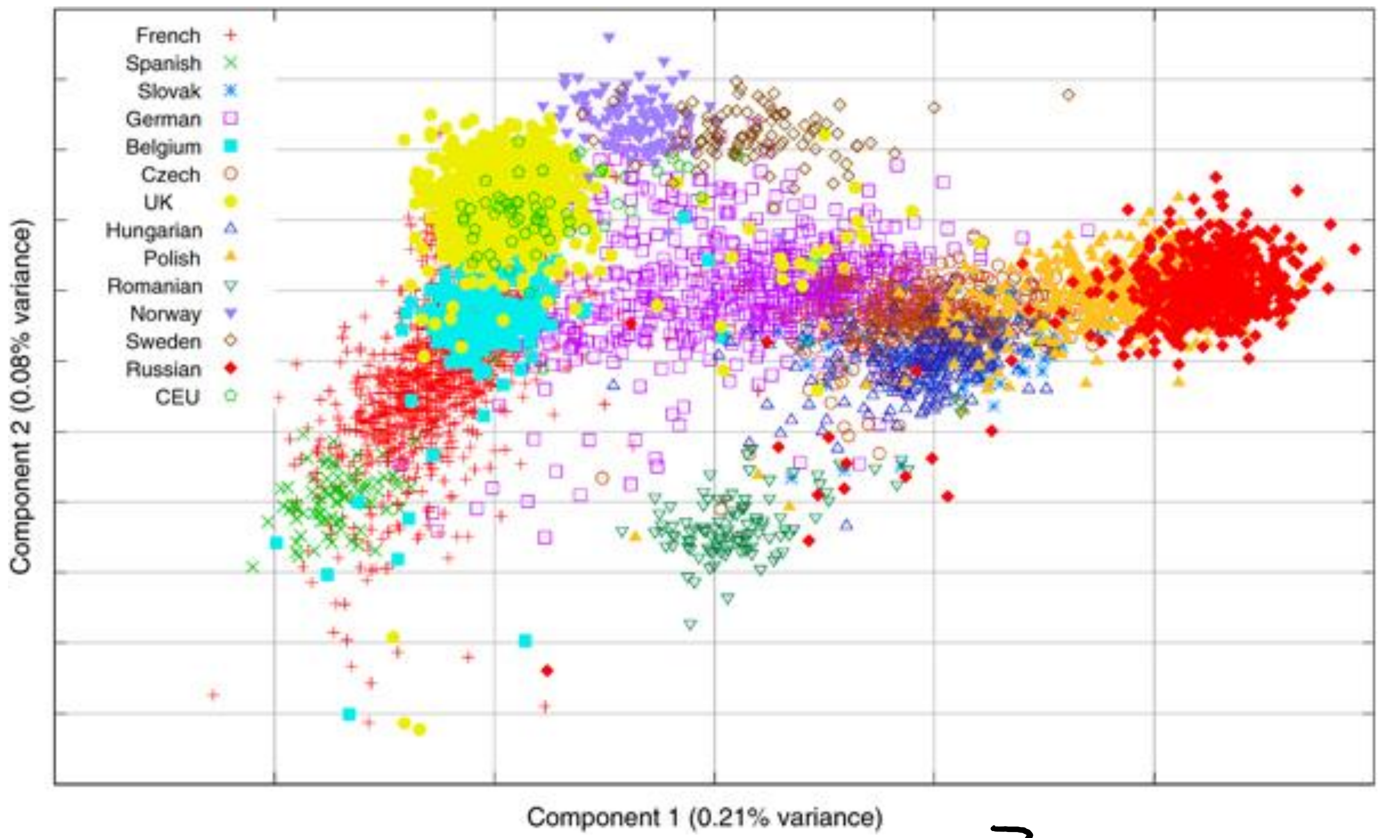
a separate 'w' trained for regression

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}

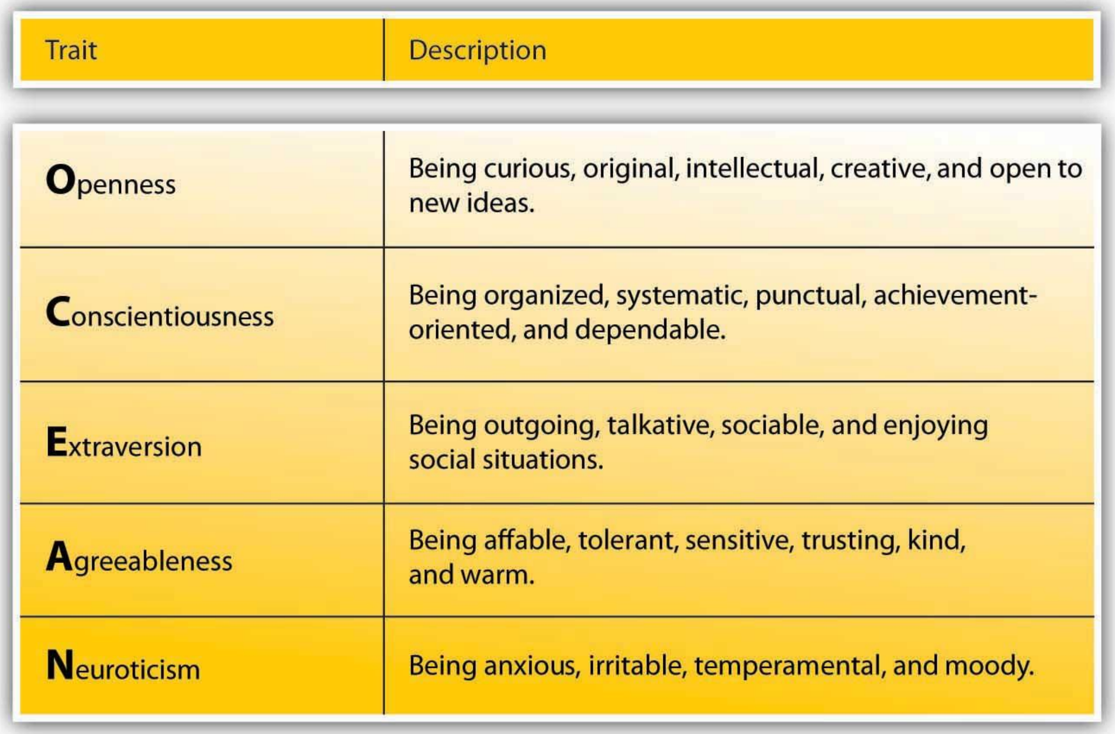


z_{i1}

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

PCA Applications

- Applications of PCA:
  - Data 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

PCA with d=1

- Consider the case of PCA when d=1:

$$X = \underset{n \times 1}{\begin{bmatrix} \\ \\ \\ \end{bmatrix}} \qquad Z = \underset{n \times 1}{\begin{bmatrix} \\ \\ \\ \end{bmatrix}} \qquad W = \underset{1 \times 1}{[\ ]}$$

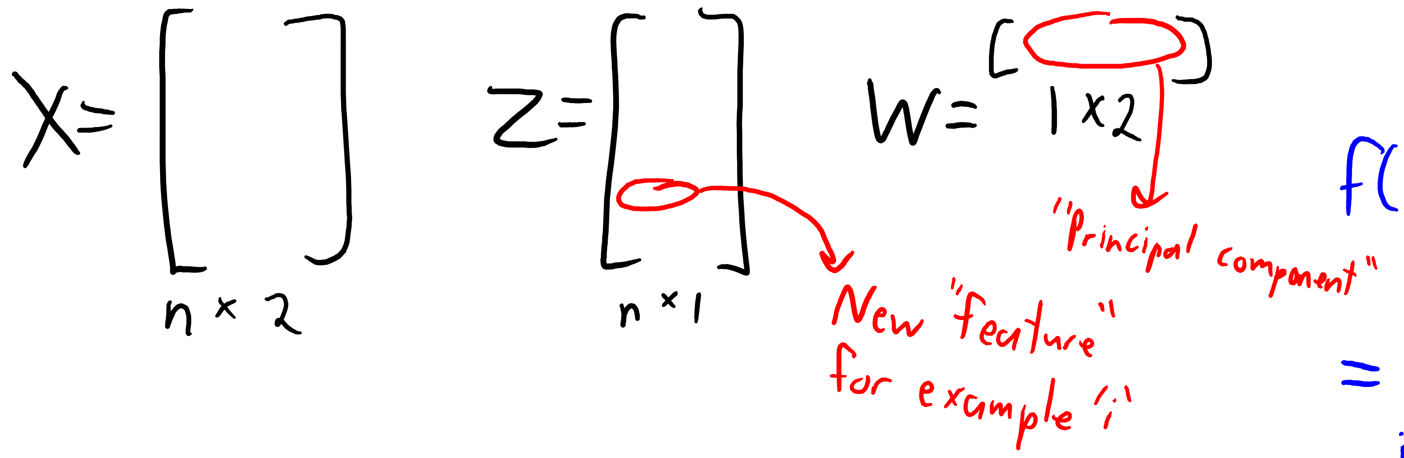
PCA objective:

$$f(Z, w) = \sum_{i=1}^{n} (w z_i - x_i)^2$$

- There is an obvious solution: w = 1 and $z_i = x_i$.
  - PCA is only interested when k < d, since otherwise we can set $z_i = x_i$.
- PCA is not unique: $w = 1/\alpha$ and $z_i = \alpha x_i$ for any 'α' is a solution.
  - We can enforce $|w| = 1$ to avoid this problem.

PCA with d=2 and k =1

- So simplest interesting case is d=2 and k=1:



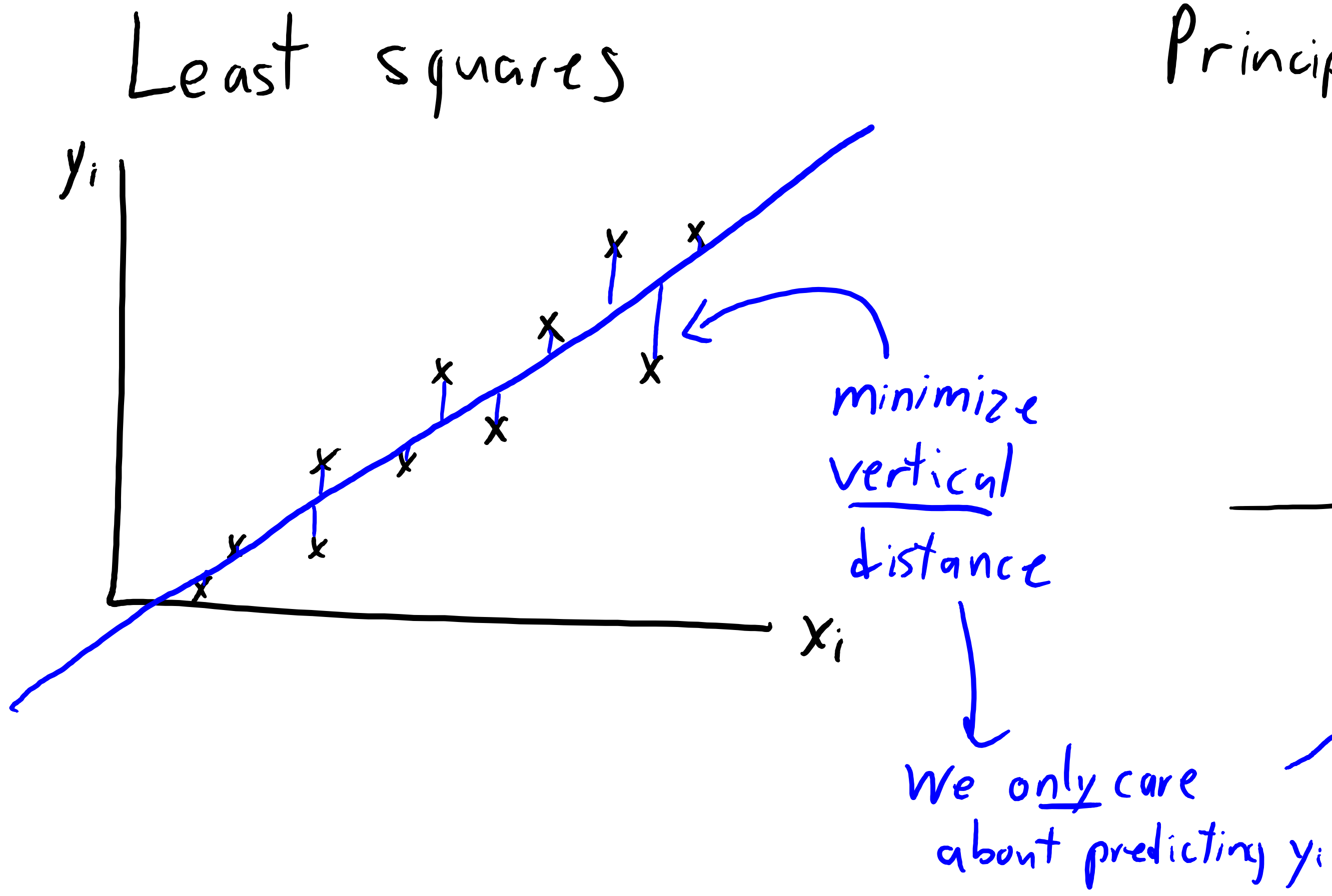
PCA objective:

$$f(Z,W) = \sum_{i=1}^{n}\sum_{j=1}^{2}(w_j z_i - x_{ij})^2$$

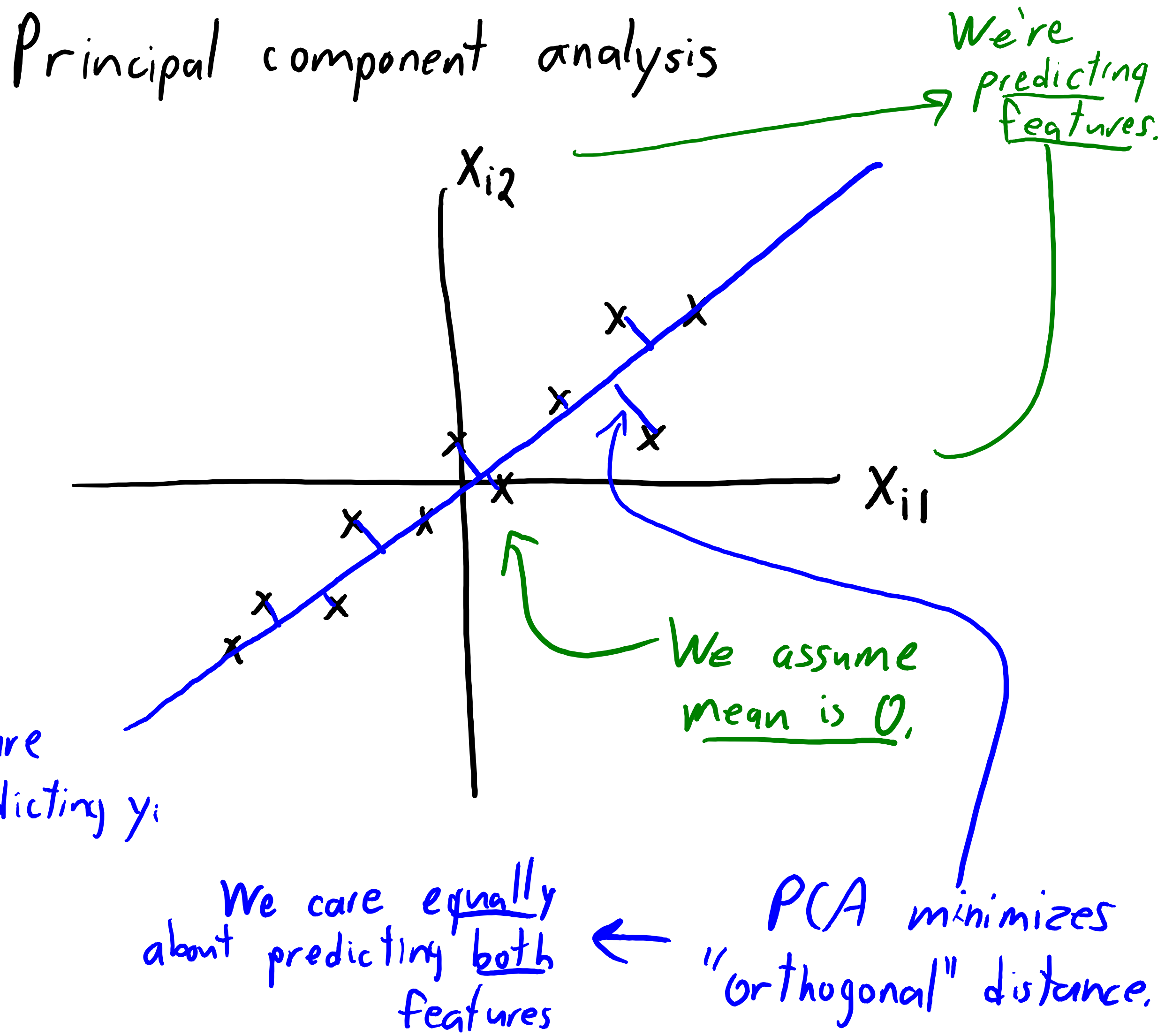
$$= \sum_{i=1}^{n}(w_1 z_i - x_{i1})^2 + \sum_{i=1}^{n}(w_2 z_i - x_{i2})^2$$

- Very similar to a least squares problem, but note that:
  - We have no 'y_i', we are trying to predict each feature x_{ij} from the single z_i.
  - But feaures 'z_i' are also variables, we are learning the features z_i too.
- Side note: in PCA we assume features have a mean of 0.
  - You can subtract mean or add bias variable if this is not true.

PCA with d=2 and k =1



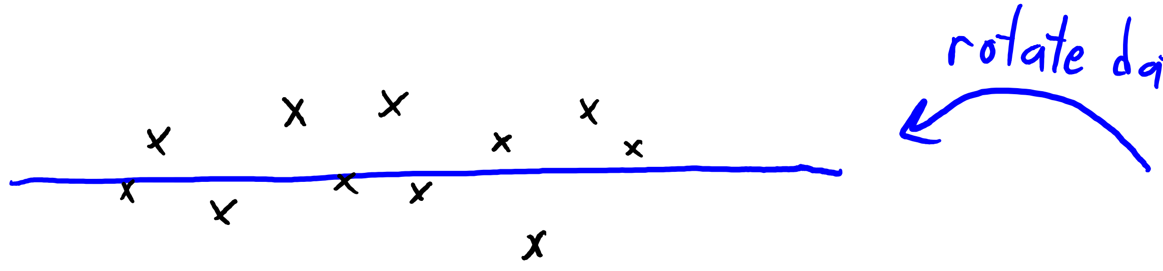
Least squares
y_i
x_i
minimize vertical distance
We only care about predicting y_i

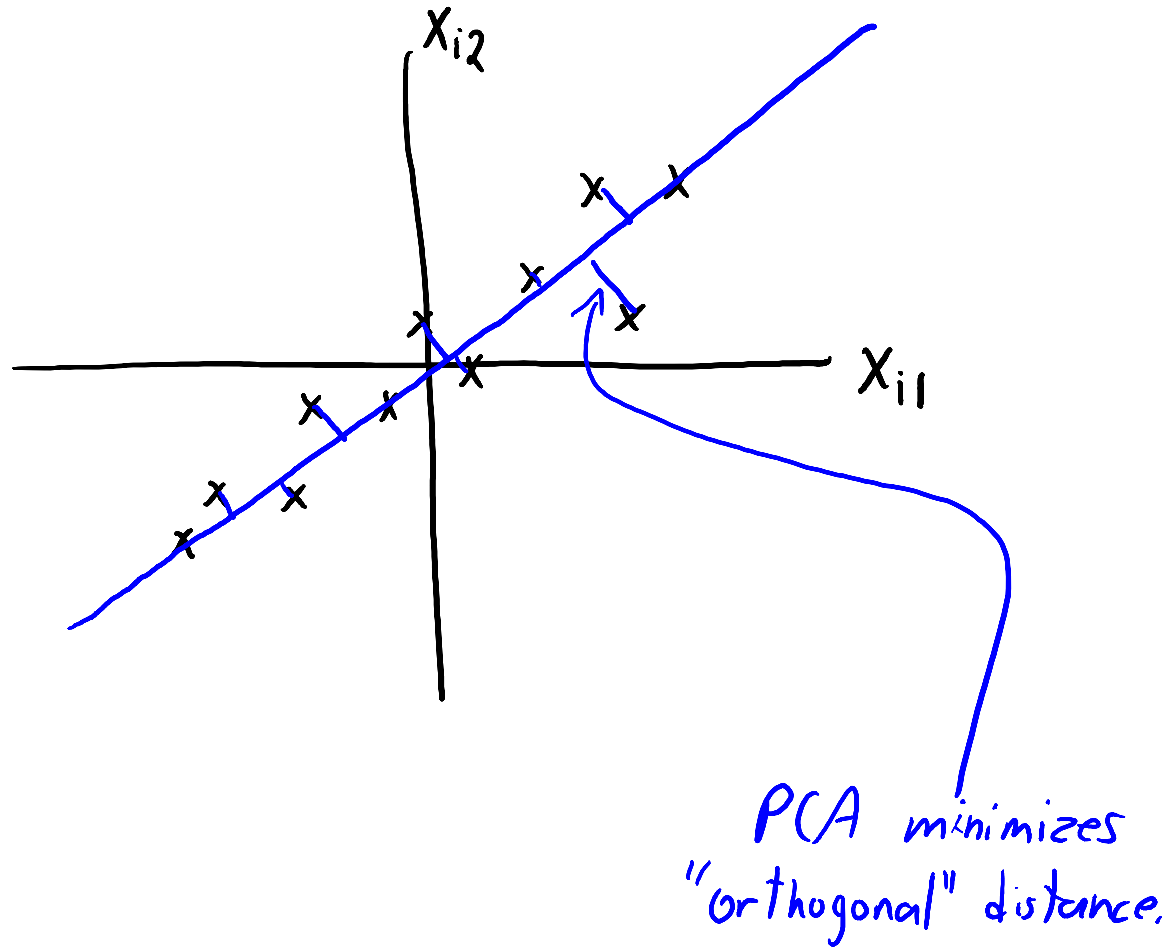


Principal component analysis
x_{i2}
x_{i1}
We're predicting features.
We assume mean is 0.
PCA minimizes "orthogonal" distance.
We care equally about predicting both features

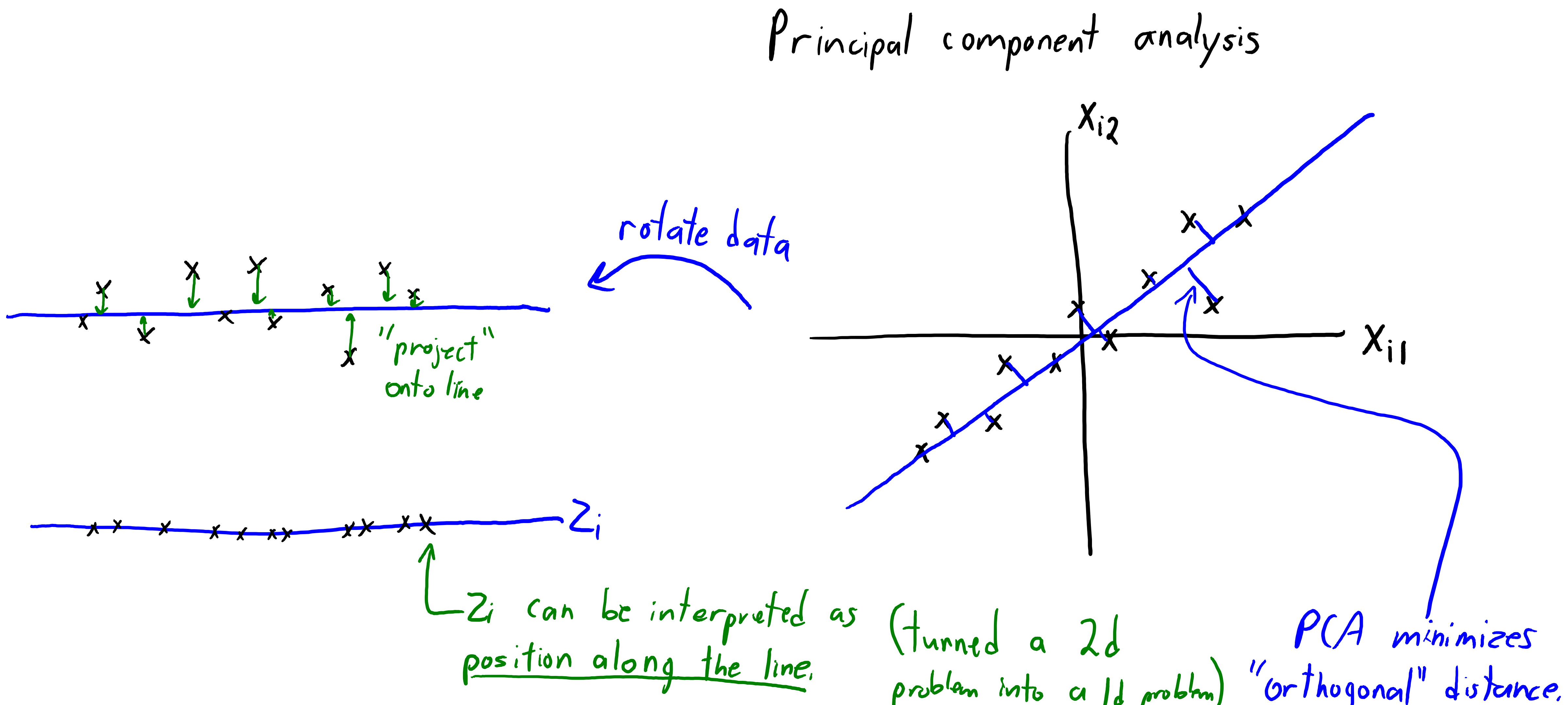
PCA with d=2 and k =1

Principal component analysis



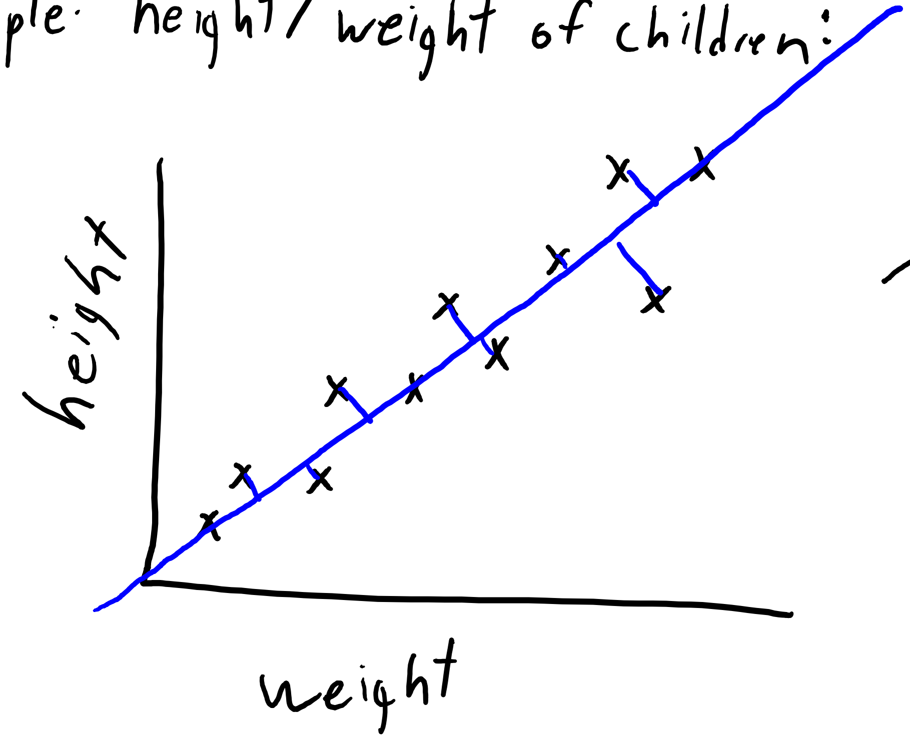


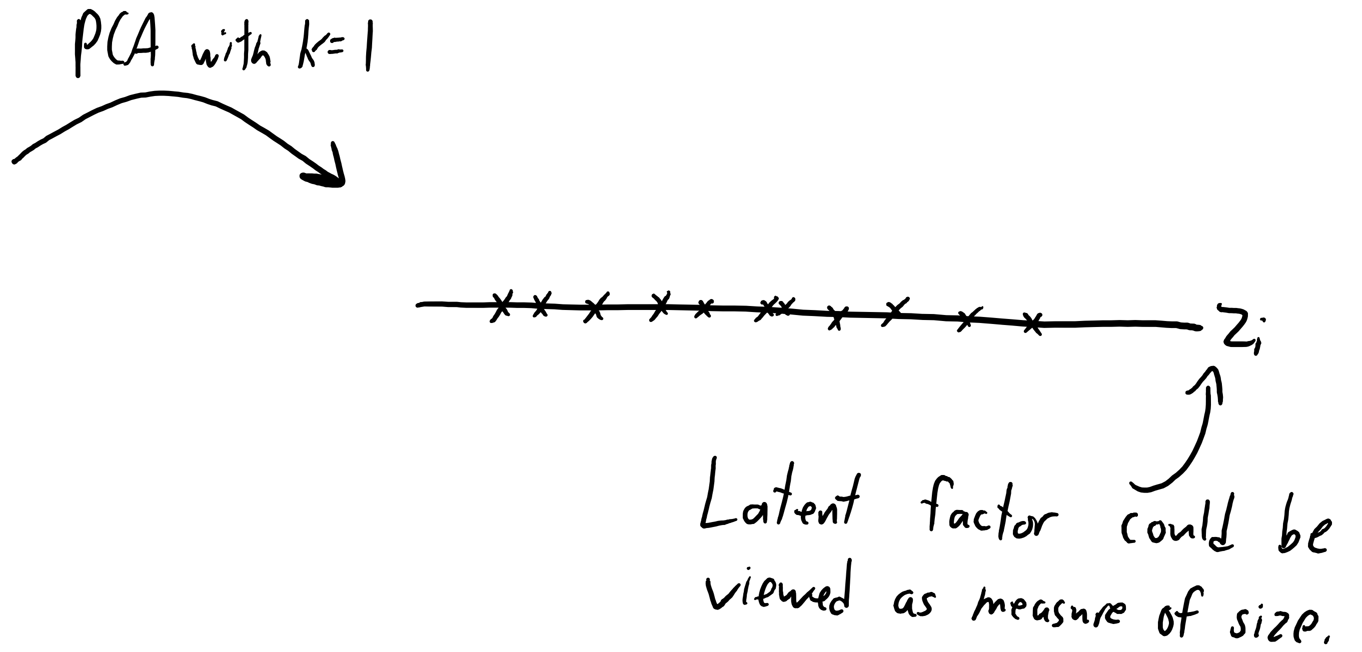
PCA with d=2 and k =1


Principal component analysis
x_{i2}
x_{i1}
rotate data
"project" onto line
z_i
z_i can be interpreted as position along the line.
(turned a 2d problem into a 1d problem)
PCA minimizes "orthogonal" distance.

PCA with d=2 and k =1

Example: height/weight of children:





PCA Computation

- The PCA objective with general ‘d’ and ‘k’:

$$f(W,Z) = \sum_{i=1}^{n}\sum_{j=1}^{d}(w_j^T z_i - x_{ij})^2$$

- 3 common ways to solve this problem:
  - Singular value decomposition: classic non-iterative approach (bonus slide).
  - Alternating minimization:
    1. Start with random initialization.
    2. Optimize ‘W’ with ‘Z’ fixed (solve gradient with respect to ‘W’ equals to 0).
    3. Optimize ‘Z’ with ‘W’ fixed (solve gradient with respect to ‘Z’ equals to 0).
    4. Go back to 2.
  - Stochastic gradient: gradient descent based on random ‘i’ and ‘j’.

PCA Non-Convexity

- The PCA objective with general ‘d’ and ‘k’:

$$f(W,Z) = \sum_{i=1}^{n}\sum_{j=1}^{d}(w_j^T z_i - x_{ij})^2$$

- This objective is not jointly convex in ‘W’ and ‘Z’.
  - This is why iterative methods need random initialization.
    - If you initialize with z1 = z2, then they stay the same.
  - But it’s possible to show that all “stable” local optima are global optima.
    - So alternating minimization and stochastic gradient give global optima in practice.

Summary

- Latent-factor models:
  - Compress data as linear combination of ‘factors’.
- Principal component analysis:
  - Most common variant based on squared reconstruction error.

- Next time: modifying PCA so it splits faces into ‘eyes’, ‘mouths’, etc.

Bonus Slide: PCA with Singular Value Decomposition

- Under constraints that $w_c^T w_c = 1$ and $w_c^T w_{c'} = 0$, use:

$$U \Sigma V^T = SVD(X)$$

$$W = V(:, 1:k)^T \qquad Z = XW^T$$

- You can also quickly get compressed version of new data:

$$\hat{Z} = \hat{X} W^T$$

- If W was not orthogonal, could get Z by least squares.