

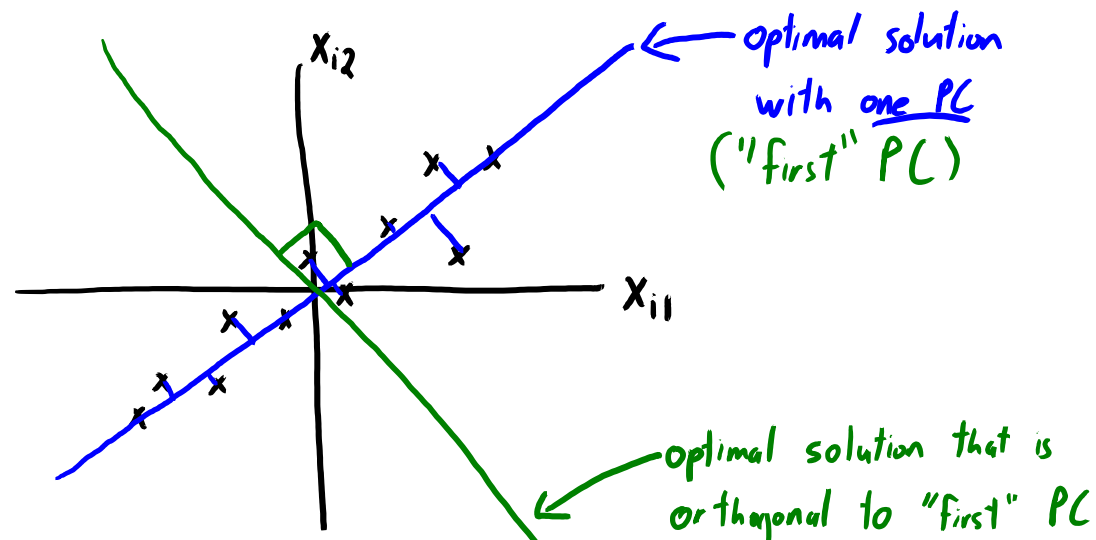
CPSC 340: Machine Learning and Data Mining

Beyond PCA

Fall 2022

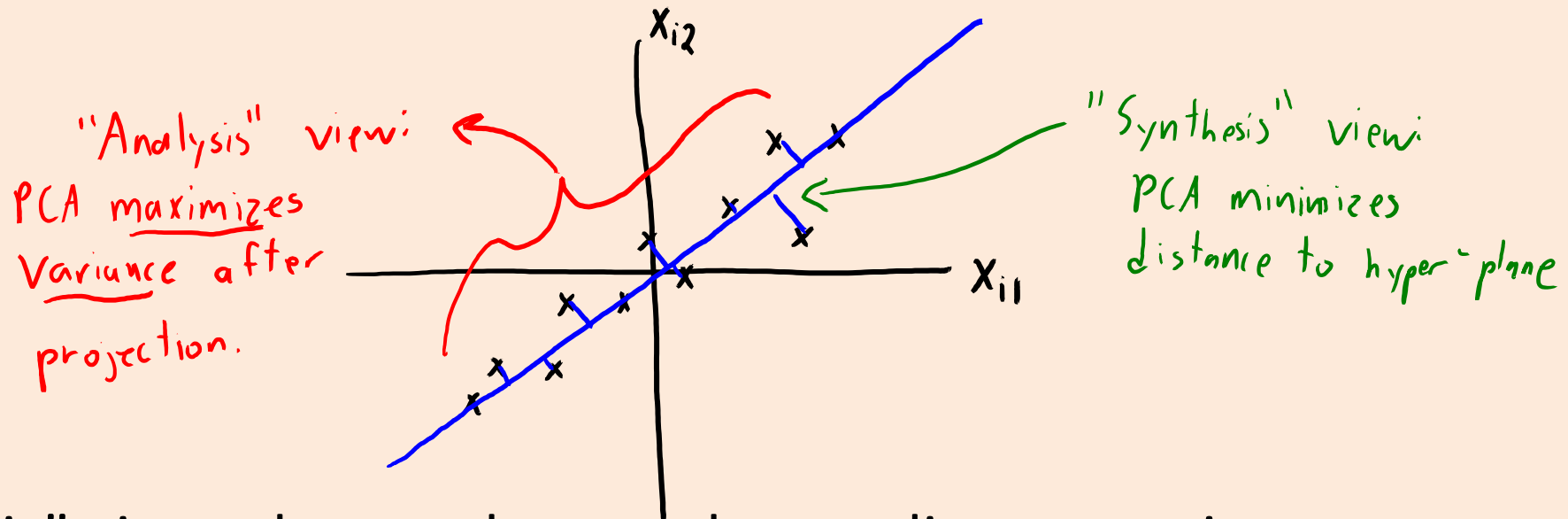
Last Time: PCA with Orthogonal/Sequential Basis

- When $k = 1$, PCA has a **scaling problem**.
- When $k > 1$, have **scaling, rotation, and label switching**.
 - Standard fix: use **normalized orthogonal rows** W_c of 'W'.
$$\|w_c\| = 1 \quad \text{and} \quad w_c^T w_{c'} = 0 \quad \text{for } c' \neq c$$
 - And **fit the rows in order**:
 - First row "reduces error the most".



“Synthesis” View vs. “Analysis” View

- We said that PCA finds hyper-plane minimizing distance to data x_i .
 - This is the “synthesis” view of PCA (connects to k-means and least squares).

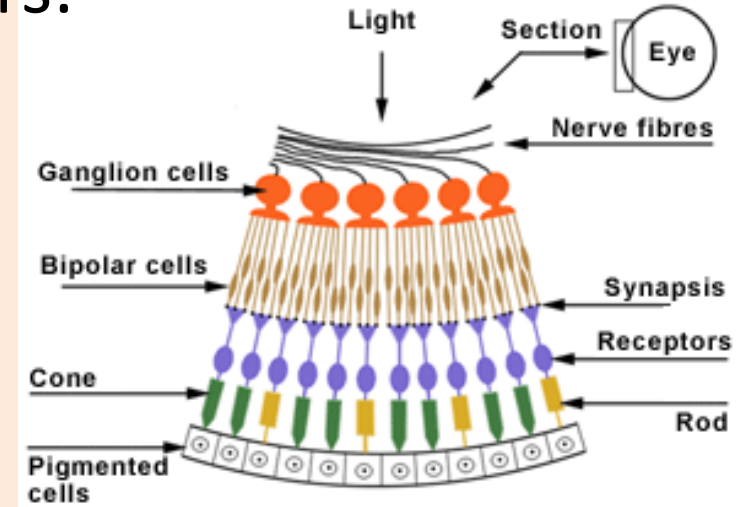


- “Analysis” view when we have orthogonality constraints:
 - PCA finds hyper-plane maximizing variance in z_i space.
 - You pick W to “explain as much variance in the data” as possible.

Colour Opponency in the Human Eye

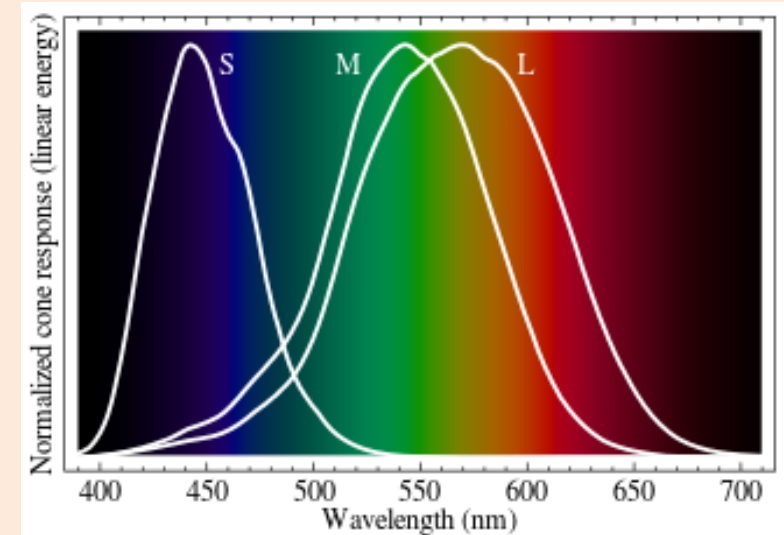
- Classic model of the eye is with 4 photoreceptors:

- Rods (more sensitive to brightness).
- L-Cones (most sensitive to red).
- M-Cones (most sensitive to green).
- S-Cones (most sensitive to blue).



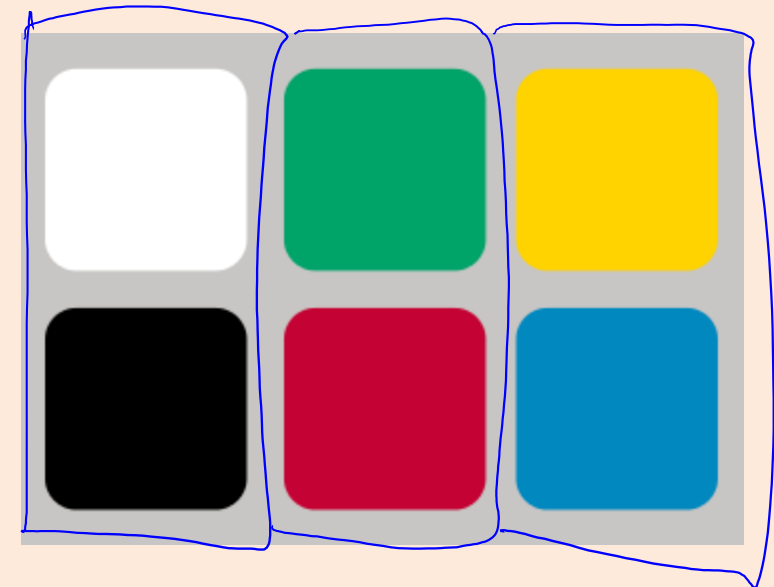
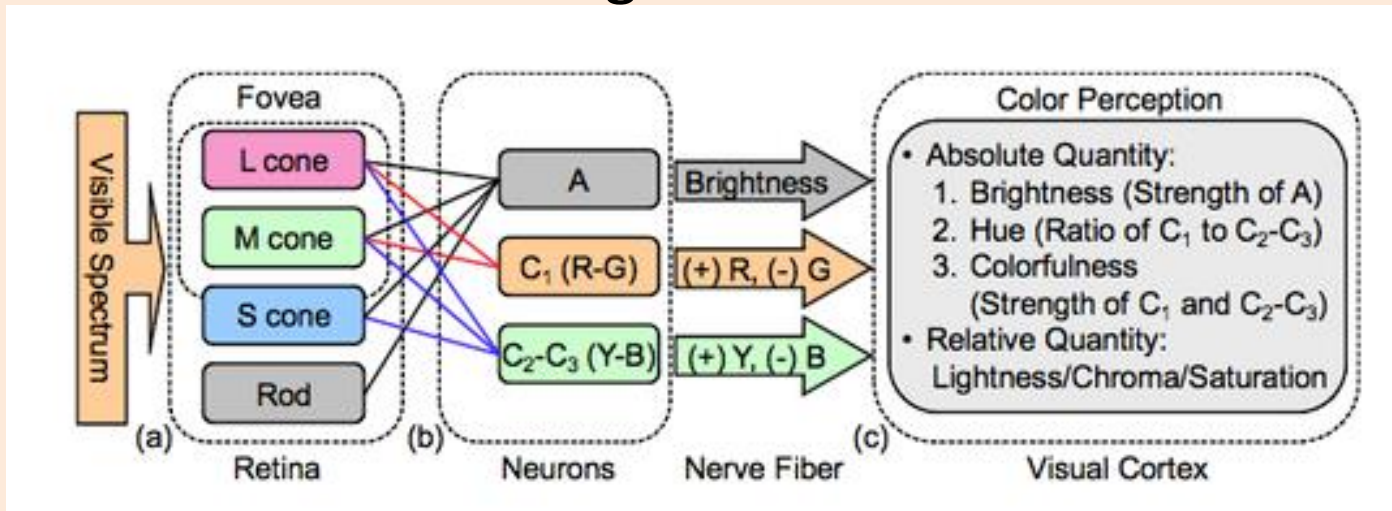
- Two problems with this system:

- Not orthogonal.
 - High correlation in particular between red/green.
- We have 4 receptors for 3 colours.

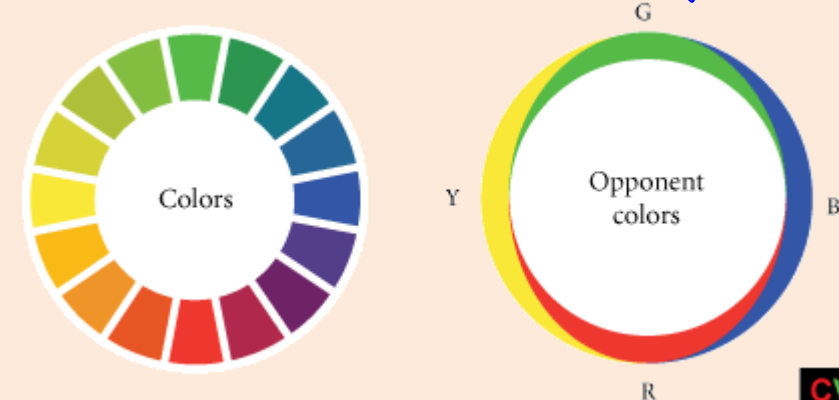


Colour Opponency in the Human Eye

- Bipolar and ganglion cells seem to code using “opponent colors”:
 - 3-variable orthogonal basis:



- This is similar to PCA ($d = 4, k = 3$).



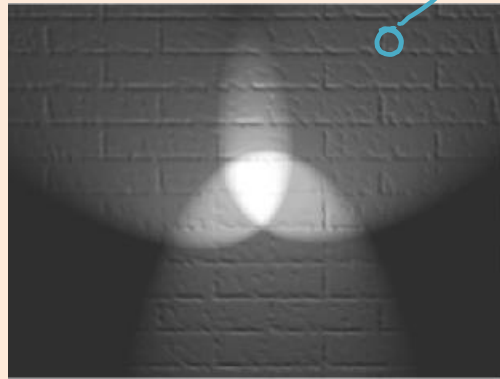
Colour Opponency Representation

For this pixel, eye gets 4 signals

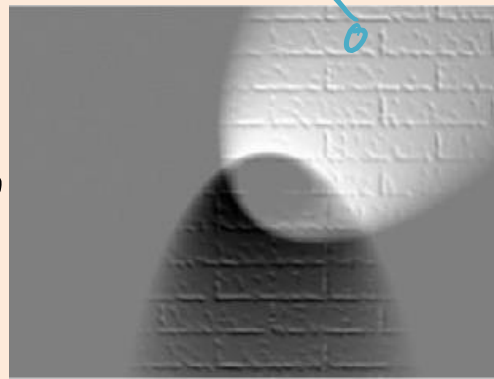
Can represent 4 original values with these 3 z_i values and matrix 'W'



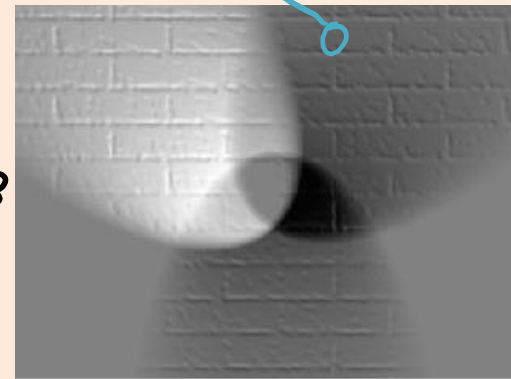
$= W_1$



$+ W_2$



$+ W_3$



First row of W (First PC)

brightness

Second row (4x1)

red/green

Third row (4x1)

blue/yellow

Analogous to means in k-means.

PCA Computation: other methods

- With **linear regression**, we had the **normal equations**
 - But we also could do it with gradient descent, SGD, etc.
- With **PCA** we have the **SVD**
 - But we can also do it with gradient descent, SGD, etc.
 - These other methods typically don't enforce the uniqueness "constraints".
 - Sensitive to initialization, don't enforce normalization, orthogonality, ordered PCs.
 - But you can do this in post-processing if you want.
 - Why would we want this? We can use our tricks from Part 3 of the course:
 - We can do things like "robust" PCA, "regularized" PCA, "sparse" PCA, "binary" PCA.
 - We can fit huge datasets where SVD is too expensive.

PCA Computation: Alternating Minimization

- With centered data, the **PCA objective** is:

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

- In **k-means** we tried to optimize this with **alternating minimization**:
 - Fix “**cluster assignments**” Z and find the optimal “**means**” W.
 - Fix “**means**” W and find the optimal “**cluster assignments**” Z.
- Converges to a local optimum.
 - But **may not find a global optimum** (sensitive to initialization).

PCA Computation: Alternating Minimization

- With centered data, the **PCA objective** is:

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

- In **PCA** we can also use **alternating minimization**:
 - Fix “**part weights**” Z and find the optimal “**parts**” W .
 - Fix “**parts**” W and find the optimal “**part weights**” Z .
- Converges to a local optimum.
 - Which will be a **global optimum** (if we randomly initialize W and Z).

PCA Computation: Alternating Minimization

- With centered data, the **PCA objective** is:

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

- **Alternating minimization** steps:

- If we fix Z , this is a quadratic function of W (least squares column-wise):

$$\nabla_W f(W, Z) = Z^T Z W - Z^T X \quad \text{so} \quad W = \underbrace{(Z^T Z)^{-1}}_{\text{(writing gradient as a matrix)}} (Z^T X)$$

- If we fix W , this is a quadratic function of Z (transpose due to dimensions):

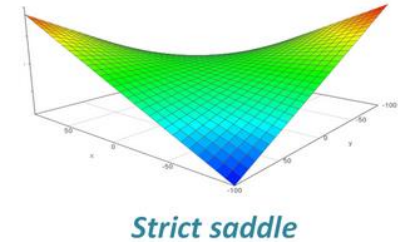
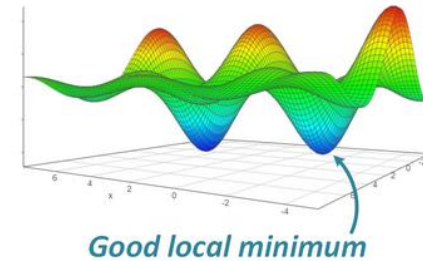
$$\nabla_Z f(W, Z) = Z W W^T - X W^T \quad \text{so} \quad Z = X W^T \underbrace{(W W^T)^{-1}}$$

These are usually invertible since $k < n$ and $k < d$

PCA Computation: Alternating Minimization

- With centered data, the **PCA objective** is:

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



- This objective is **not jointly convex** in W and Z .
 - You will **find different W and Z depending on the initialization.**
 - For example, if you initialize with all $w_c = 0$, then they will stay at zero.
 - But it's possible to show that **all "stable" local optima are global optima.**
 - You will **converge to a global optimum in practice** if you **initialize randomly.**
 - Randomization means you don't start on one of the unstable non-global critical points.
 - E.g., sample each initial z_{ij} from a normal distribution.

PCA Computation: Stochastic Gradient Descent

- For big X matrices, you can also use **stochastic gradient descent**:

$$f(W, z) = \sum_{i=1}^n \sum_{j=1}^d (\langle w^j, z_i \rangle - x_{ij})^2 = \sum_{(i,j)} (\underbrace{\langle w^j, z_i \rangle - x_{ij}}_{f(w^j, z_i, x_{ij})})^2$$

On each iteration, pick a random example i and feature j :

$$\rightarrow \text{Set } w^j \text{ to } w^j - \alpha^t \nabla_{w^j} f(w^j, z, x_{ij})$$

$$\rightarrow \text{Set } z_i \text{ to } z_i - \alpha^t \nabla_{z_i} f(w^j, z_i, x_{ij})$$

- Other variables stay the same, **cost per iteration is only $O(k)$** .

Next Topic: Variations on PCA

Beyond Squared Error

- Our objective for **latent-factor models** (LFM):

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

- As before, there are **alternatives to squared error**.

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d \text{loss}(\langle w^j, z_i \rangle, x_{ij})$$

*Error for predicting $\langle w^j, z_i \rangle$
when true value is x_{ij}*

- If X has of +1 and -1 values, we could use the **logistic loss**:

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d \log(1 + \exp(-x_{ij} \langle w^j, z_i \rangle))$$

- And predict x_{ij} with $\text{sign}(\langle w^j, z_i \rangle)$, which would be a **binary PCA** model.

Beyond Squared Error

- Our objective for **latent-factor models** (LFM):

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

- As before, there are **alternatives to squared error**.

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d \text{loss}(\langle w_j^i, z_i \rangle, x_{ij})$$

Error for predicting $\langle w_j^i, z_i \rangle$ when true value is x_{ij}

- If X has many outliers, we could use the **absolute loss**:

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

- Which will be robust to outliers and is called **robust PCA** model.

Regularized Matrix Factorization

- Recently people have also considered **L2-regularized PCA**:

$$f(W, Z) = \frac{1}{2} \|ZW - X\|_F^2 + \frac{\lambda_1}{2} \|W\|_F^2 + \frac{\lambda_2}{2} \|Z\|_F^2$$

- **Replaces normalization/orthogonality/sequential-fitting.**
 - Often gives **lower reconstruction error on test data.**
 - But requires **regularization parameters** λ_1 and λ_2 .
- **Need to regularize W and Z** because of scaling problem.
 - **If you only regularize 'W' it does not do anything.**
 - I could take unregularized solution, replace W by αW for a tiny α to shrink $\|W\|_F$ as much as I want, then multiply Z by $(1/\alpha)$ to get same solution.
 - Similarly, **if you only regularize 'Z' it does not do anything.**

Sparse Matrix Factorization and NMF

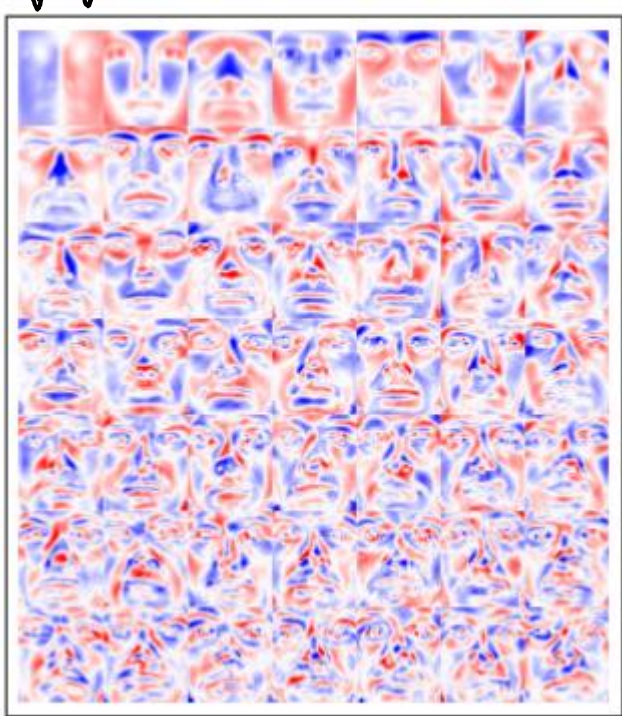
- Alternately, many works consider **L1-regularization**:

$$f(W, Z) = \frac{1}{2} \|ZW - X\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^n \|z_i\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^d \|w_j\|_1$$

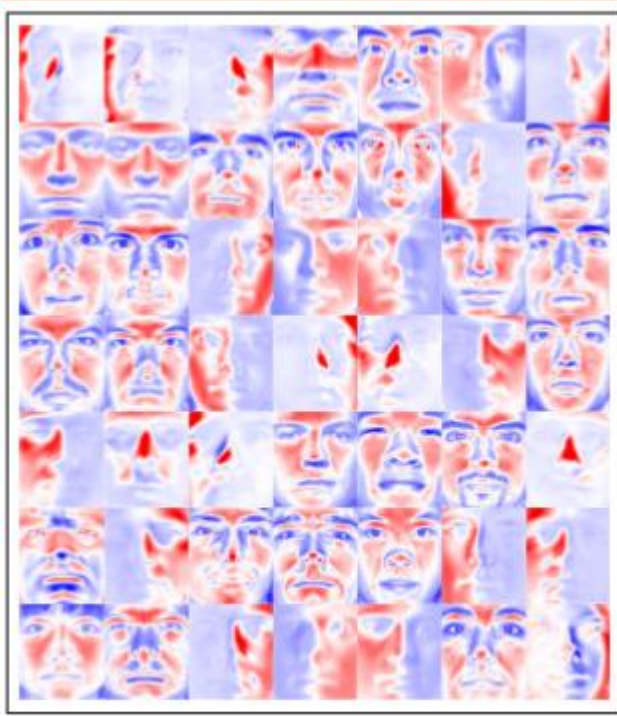
- Called **sparse coding** (L1 on ‘Z’) or **sparse dictionary learning** (L1 on ‘W’).
 - Encourage values of ‘Z’ or ‘W’ to be exactly **zero** as we increase λ_1 or λ_2 .
- A related older method is **non-negative matrix factorization (NMF)**:
 - Optimizes the PCA objective but forces ‘Z’ and ‘W’ to be **non-negative**.
 - In some applications negative quantities do not make sense.
 - The non-negative constraint also leads to **sparsity** (many values set to 0).
 - But unlike L1-regularization, you **cannot control the degree of sparsity**.
 - Details on NMF in bonus (including “cancer signatures” and “NBA shot charts”).

Sparse Eigenfaces

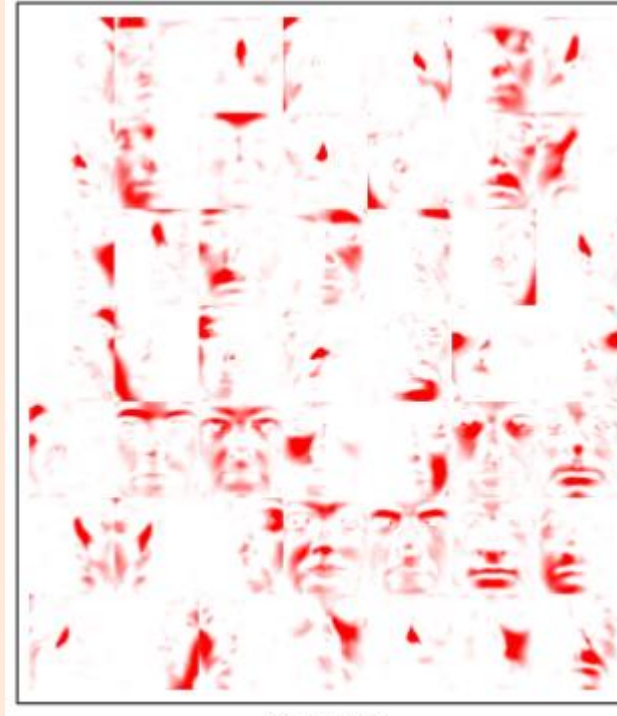
blue: negative
red: positive



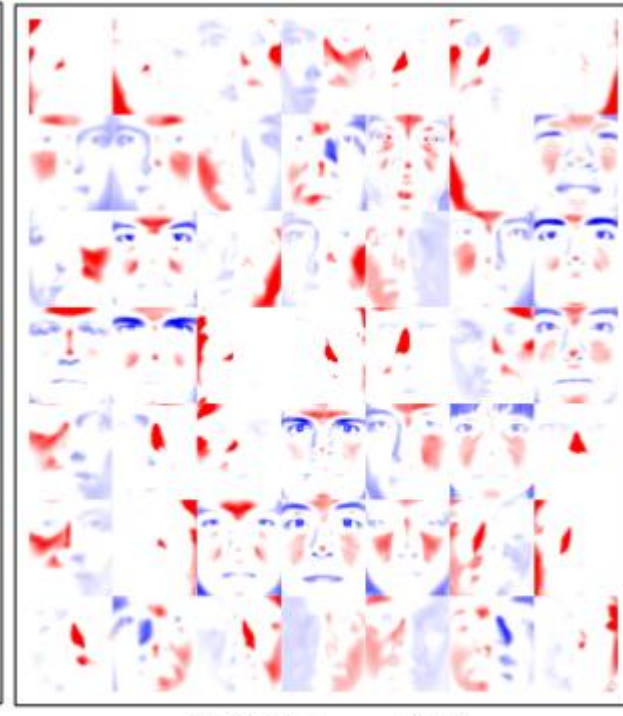
(a) PCA



(e) Dictionary Learning



(c) NMF



(d) SPCA, $\tau = 30\%$

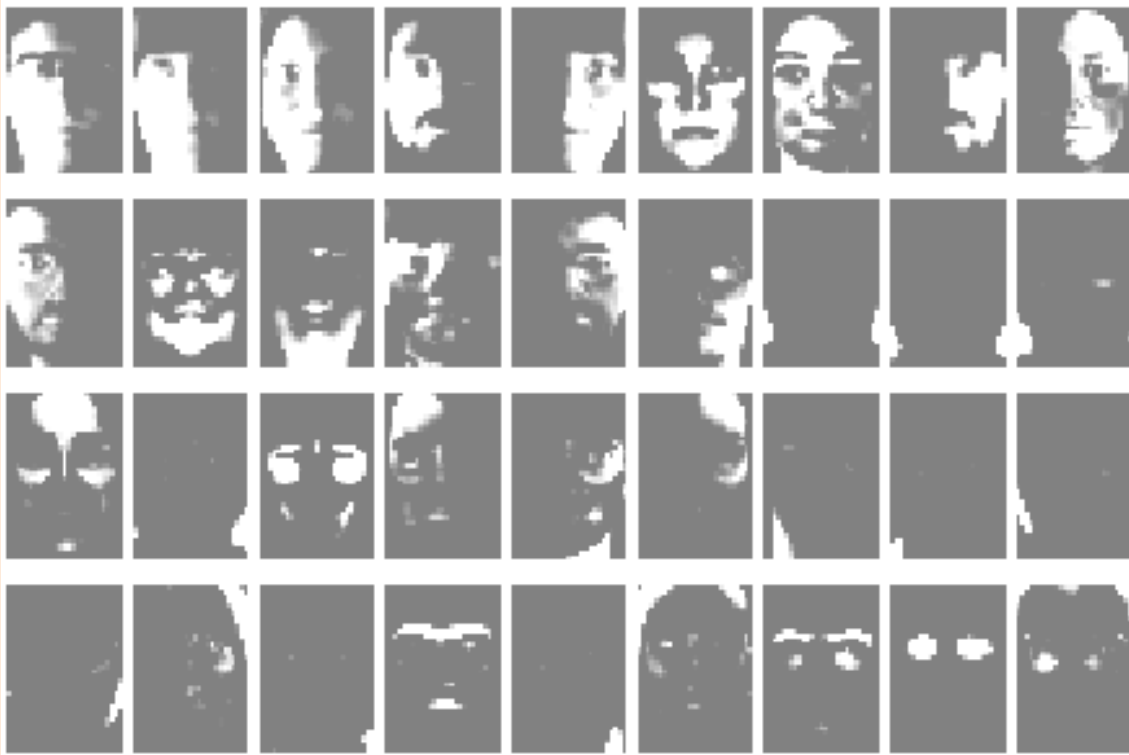
PCA without orthogonality

sparsity due to non-negativity

sparsity due to L_1 -regularization

Recent Work: Structured Sparsity

- “**Structured sparsity**” considers dependencies in sparsity patterns.
 - Can enforce that “parts” are convex regions.

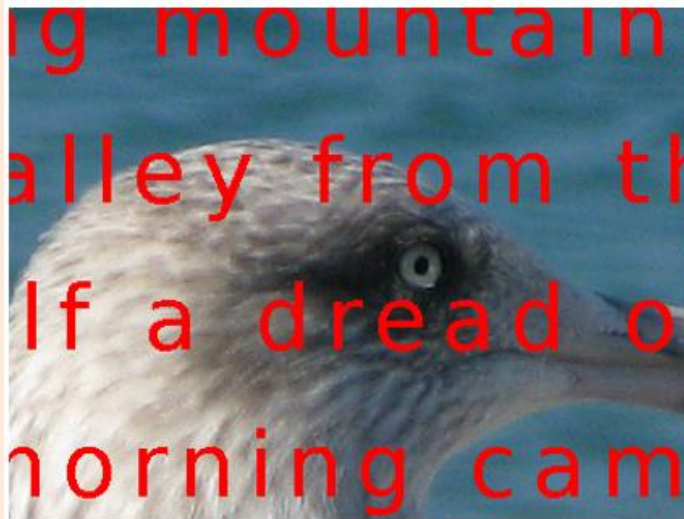


NMF



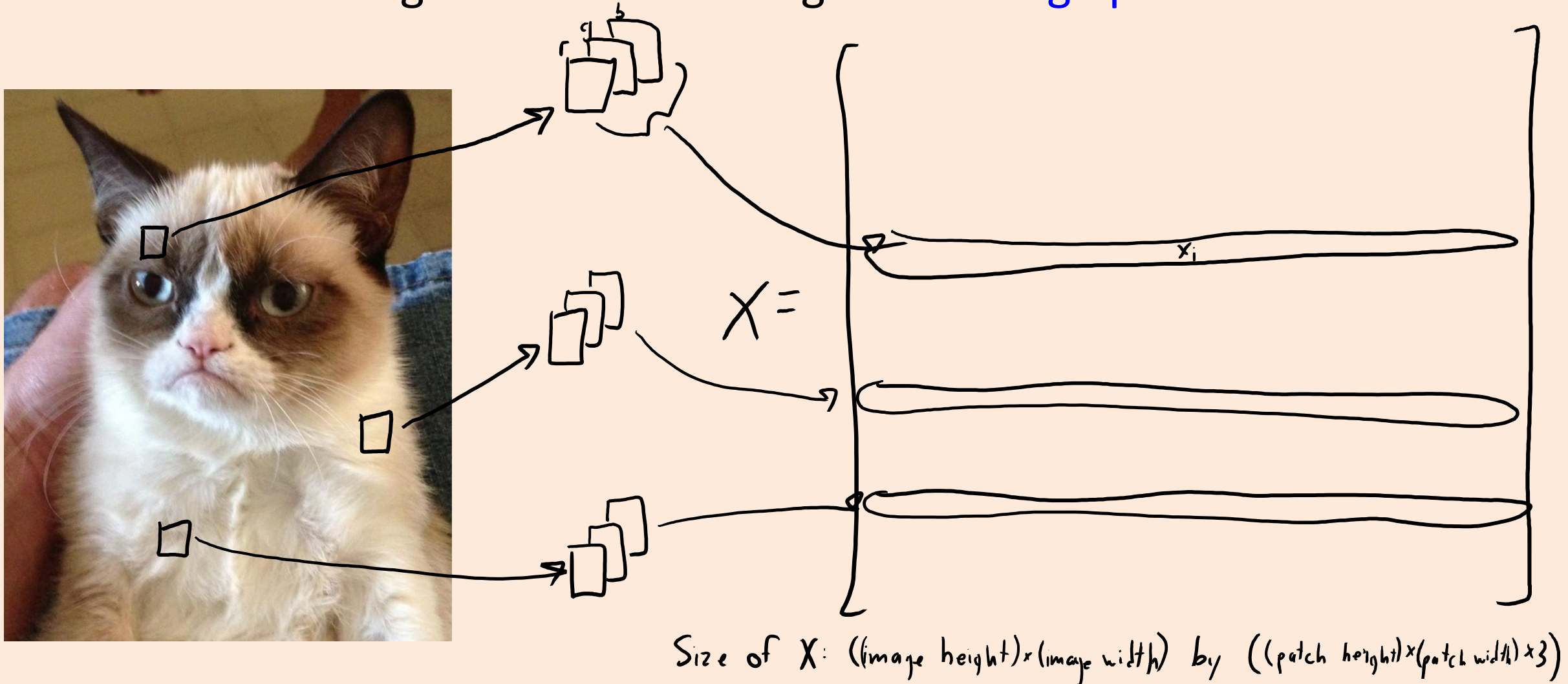
Sparse PCA with “structured” sparsity

Application: Image Restoration



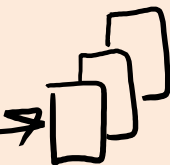
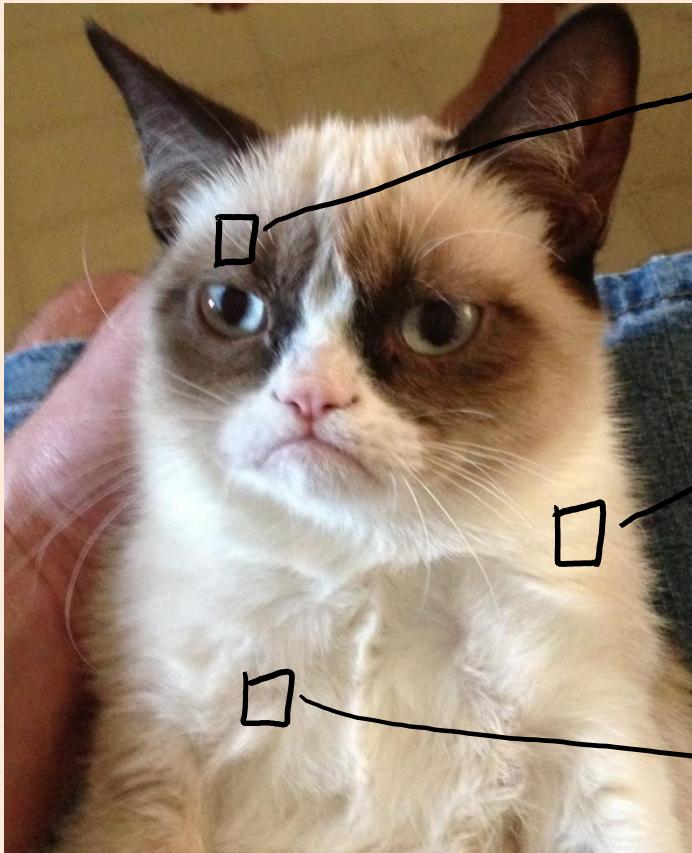
Latent-Factor Models for Image Patches

- Consider building latent-factors for general **image patches**:



Latent-Factor Models for Image Patches

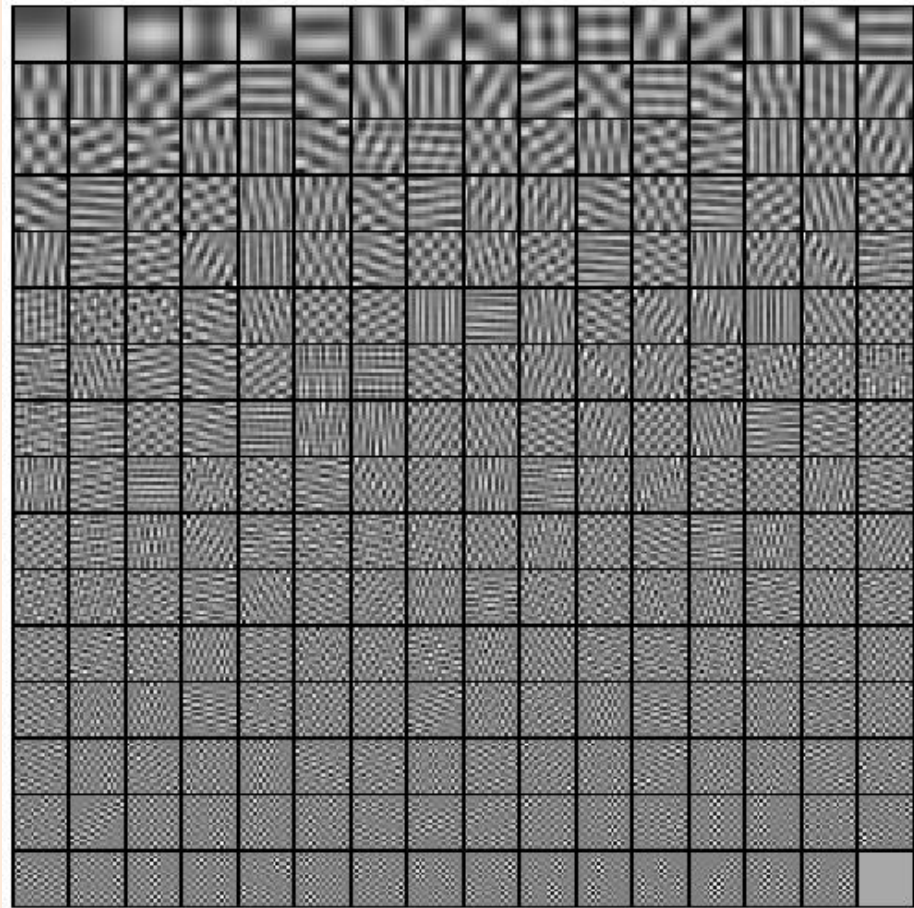
- Consider building latent-factors for general **image patches**:



Typical pre-processing:

1. Usual variable centering
2. “Whiten” patches.
(remove correlations - bonus)

Latent-Factor Models for Image Patches

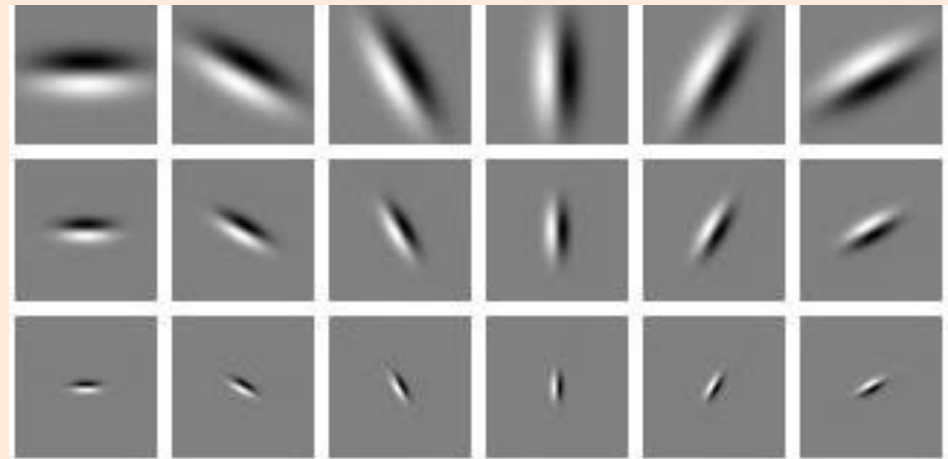


(b) Principal components.

Orthogonal bases don't seem right:

- Few PCs do almost everything.
- Most PCs do almost nothing.

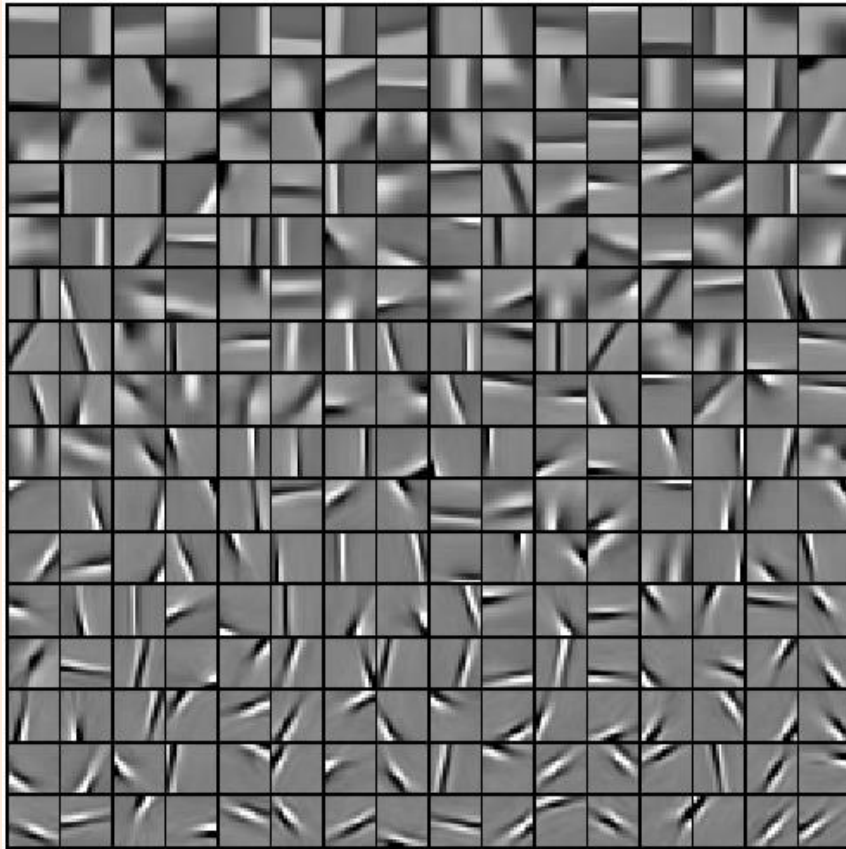
We believe “simple cells” in visual cortex use:



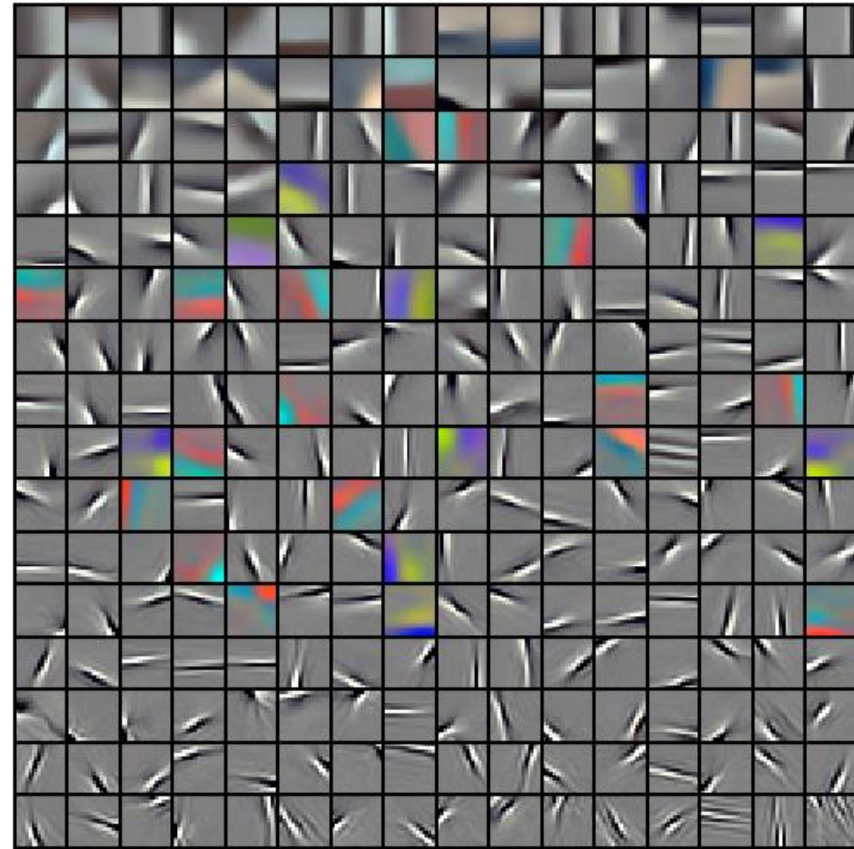
'Gabor' filters

Latent-Factor Models for Image Patches

- Results from a non-orthogonal latent factor with L1-regularization:



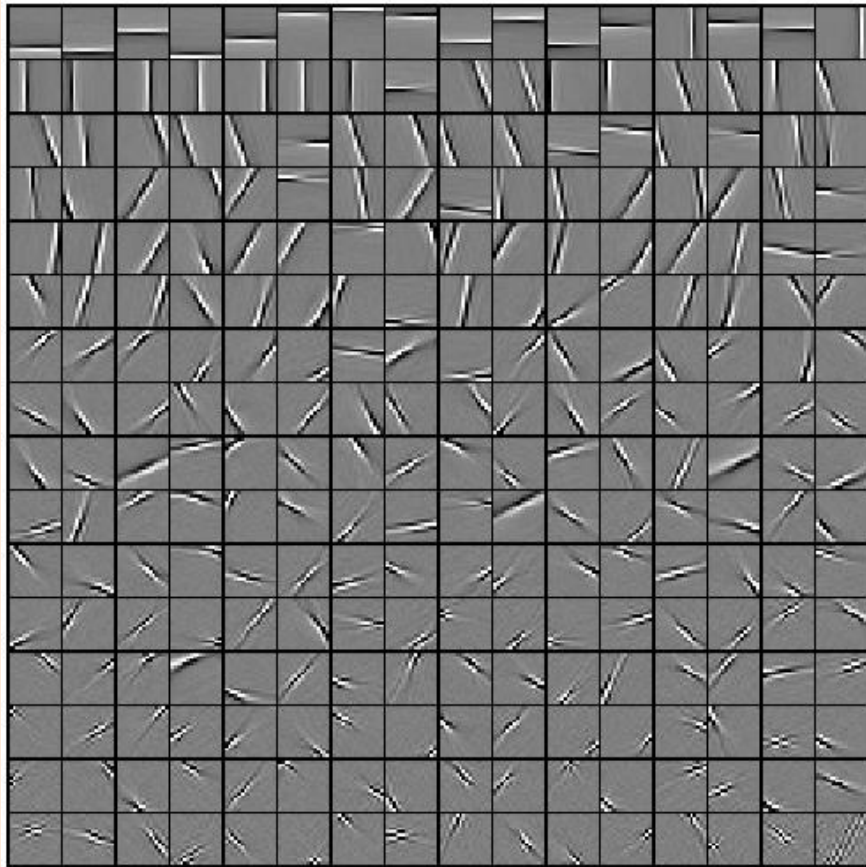
(a) With centering - gray.



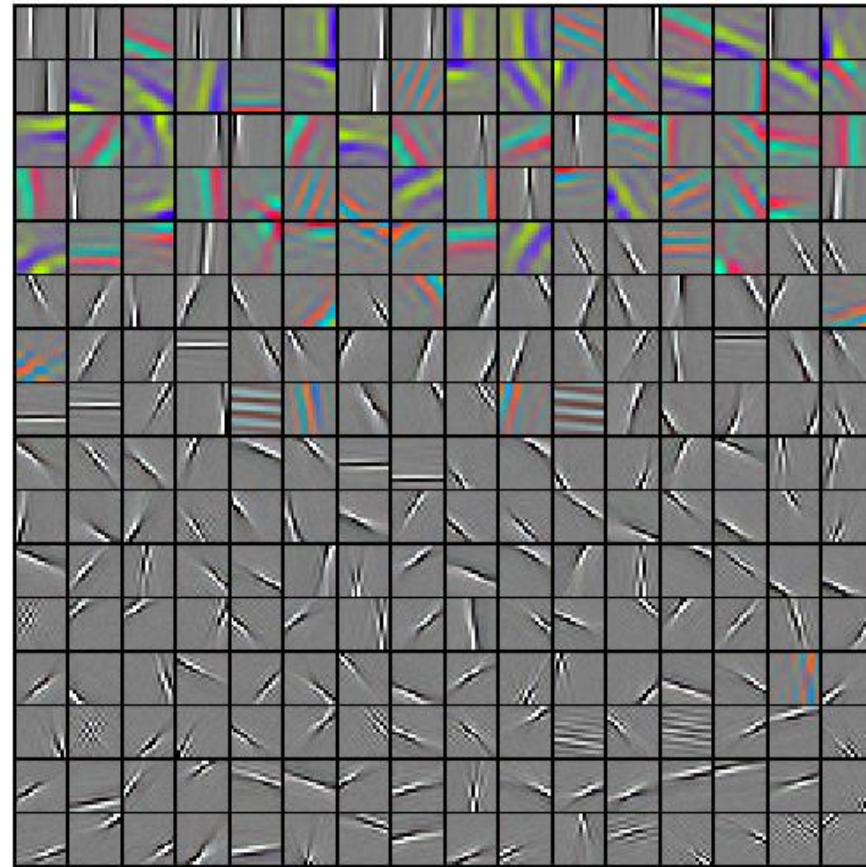
(b) With centering - RGB.

Latent-Factor Models for Image Patches

- Results from a non-orthogonal latent factor with L1-regularization:



(c) With whitening - gray.

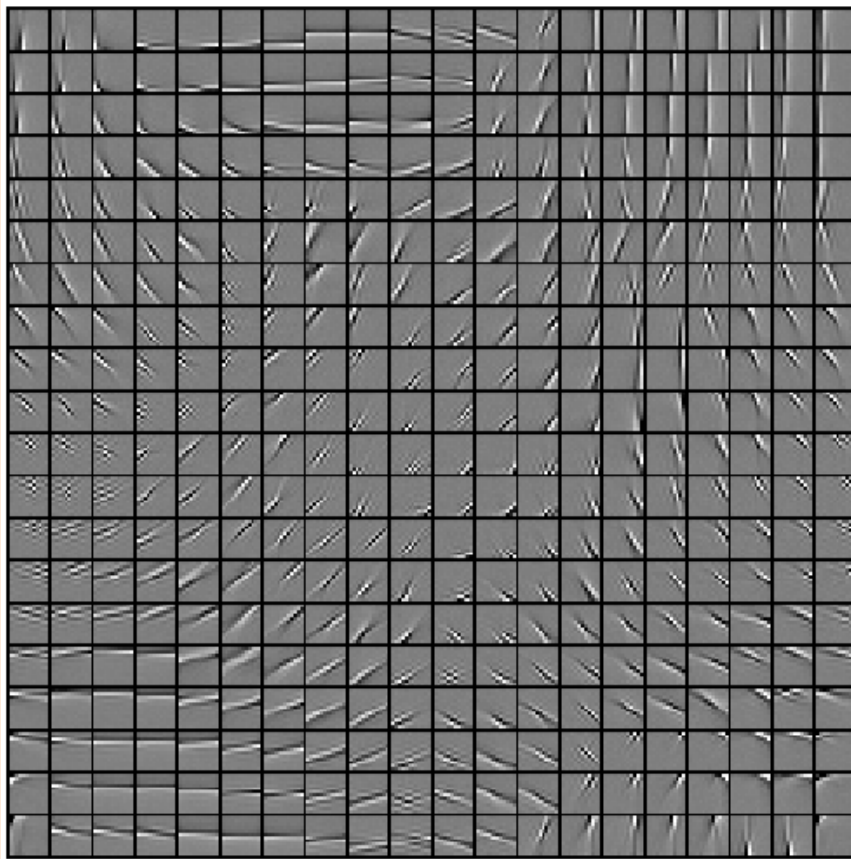


(d) With whitening - RGB.

"colour opponency"

Recent Work: Structured Sparsity

- Basis learned with a “structured sparsity” regularizer:



(b) With 4×4 neighborhood.

Similar to "cortical columns"
theory in visual cortex.

Next Topic: Recommender Systems

Recommender System Motivation: Netflix Prize

- Netflix Prize:
 - 100M ratings from 0.5M users on 18k movies.
 - Grand prize was \$1M for first team to reduce squared error by 10%.
 - Started on October 2nd, 2006.
 - Netflix's system was first beat October 8th.
 - 1% error reduction achieved on October 15th.
 - Steady improvement after that.
 - ML methods soon dominated.

Motivation: Other Recommender Systems

- Recommender systems are now everywhere:
 - Music, news, books, jokes, experts, restaurants, friends, dates, etc.
- Main types of approaches:
 1. Content-based filtering.
 - Supervised learning:
 - Extract features x_i of users and items, building model to predict rating y_i given x_i .
 - Apply model to prediction for new users/items.
 - Example: G-mail's "important messages" (personalization with "local" features).
 2. Collaborative filtering.
 - "Unsupervised" learning (have label matrix 'Y' but no features):
 - We only have labels y_{ij} (rating of user 'i' for movie 'j').
 - Example: Amazon recommendation algorithm.

Lessons Learned from Netflix Prize

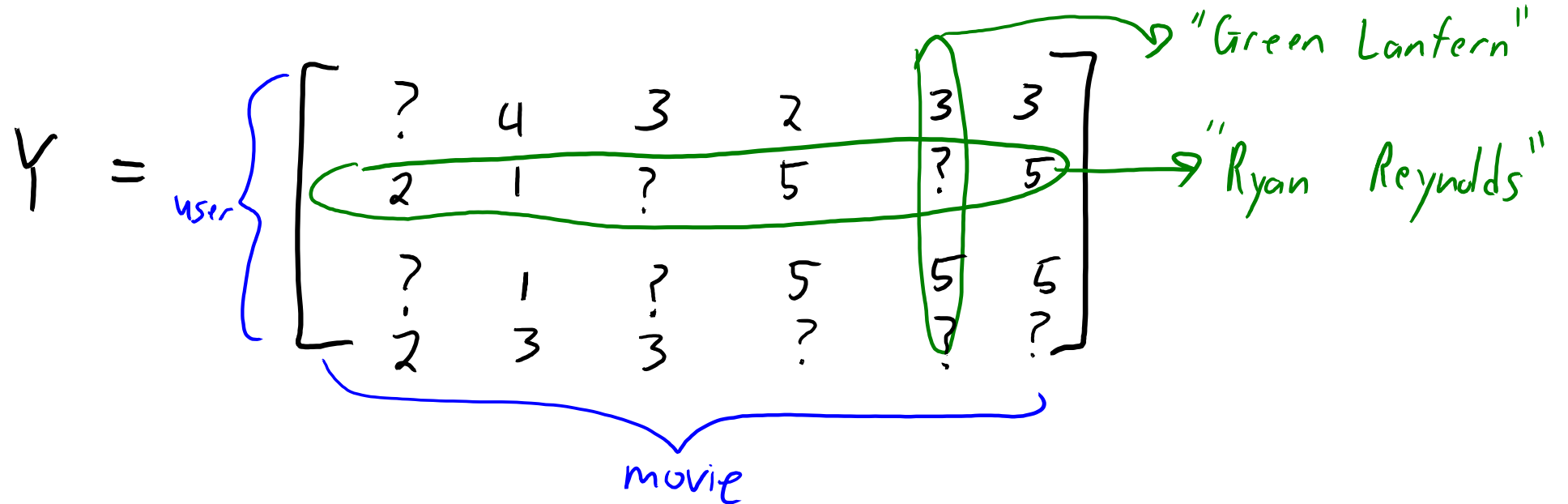
- Netflix prize awarded in 2009:
 - Ensemble method that averaged 107 models.
 - Increasing diversity of models more important than improving models.



- Winning entry (and most entries) used collaborative filtering:
 - Methods that only looks at ratings, not features of movies/users.
- A simple collaborative filtering method that does really well (7%):
 - “Regularized matrix factorization”. Now adopted by many companies.

Collaborative Filtering Problem

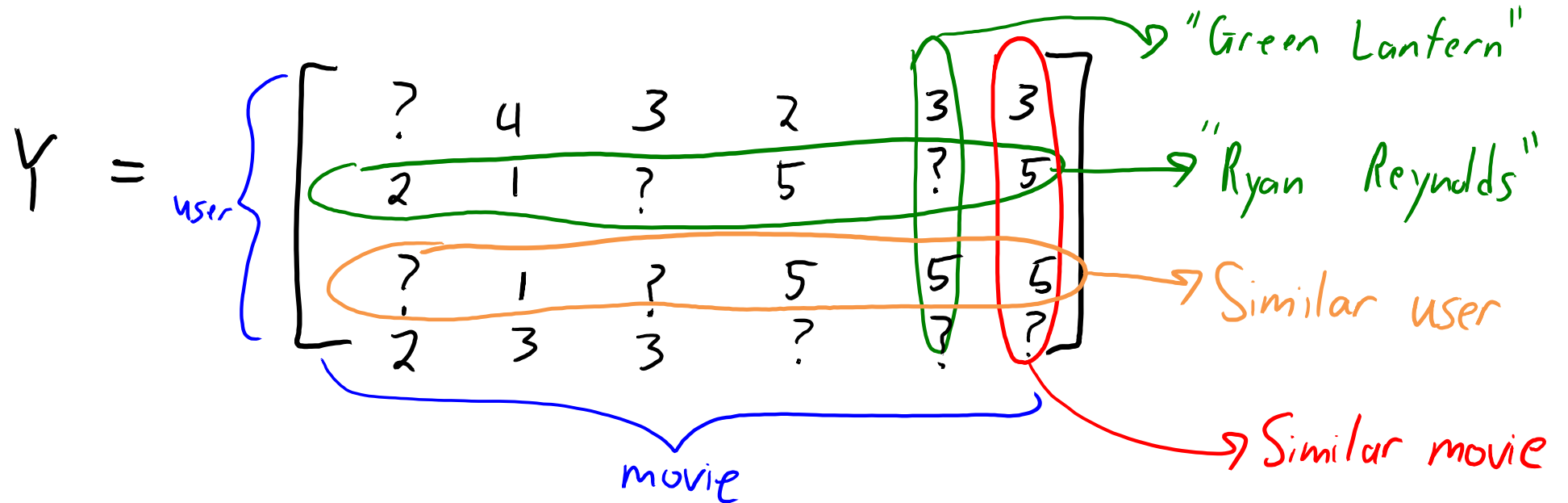
- Collaborative filtering is 'filling in' the **user-item matrix**:



- We have some ratings available with values {1,2,3,4,5}.
- We want to **predict ratings “?”** by looking at available ratings.

Collaborative Filtering Problem

- Collaborative filtering is 'filling in' the **user-item matrix**:



- What rating would "Ryan Reynolds" give to "Green Lantern"?
 - Why is this not completely crazy? We may have **similar users and movies**.

Matrix Factorization for Collaborative Filtering

- Our standard **latent-factor model** for entries in matrix 'Y':

$$Y \approx ZW$$

$n \times d$ $n \times k$ $k \times d$

$$y_{ij} \approx \langle w^j, z_i \rangle$$

- User 'i' has latent features z_i .

z_{ic} could mean "likes Nicolas Cage"

- Movie 'j' has latent features w^j .

w_{jc} could mean "has Nicolas Cage"

- Our loss function sums over **available ratings 'R'**:

$$f(Z, w) = \sum_{(i,j) \in R} (\langle w^j, z_i \rangle - y_{ij})^2 + \frac{\lambda_1}{2} \|Z\|_F^2 + \frac{\lambda_2}{2} \|W\|_F^2$$

- And we add **L2-regularization** to both types of features.
 - Basically, this is **regularized PCA on the available entries of Y**.
 - Typically fit with **SGD**.
- This simple method gives you a 7% improvement on the Netflix problem.

Adding Global/User/Movie Biases

- Our standard **latent-factor model** for entries in matrix 'Y':

$$\hat{y}_{ij} = \langle w^j, z_i \rangle$$

- Sometimes we **don't assume the y_{ij} have a mean of zero:**

- We could add bias β reflecting average overall rating:

$$\hat{y}_{ij} = \beta + \langle w^j, z_i \rangle$$

- We could also add a **user-specific bias β_i** and **item-specific bias β_j** .

$$\hat{y}_{ij} = \beta + \beta_i + \beta_j + \langle w^j, z_i \rangle$$

- Some users rate things higher on average, and movies are rated better on average.
- These might also be regularized.

Summary

- **Biological motivation** for orthogonal and/or sparse latent factors.
- **Alternating minimization and stochastic gradient descent:**
 - Iterative algorithms for minimizing PCA objective.
- Many of our regression tricks can be used with LFMs:
 - **Robust PCA** uses absolute error to be robust to outliers.
 - **Regularized PCA** can improve generalization or give sparse factors.
- **Recommender systems** try to recommend products.
- **Collaborative filtering** tries to fill in missing values in a matrix.
 - **Matrix factorization** is a common approach (“PCA with missing entries”).
- Next time: should we make a scatterplot with gradient descent?

Proof: "Synthesis" View = "Analysis" View ($WW^T = I$)

- The **variance of the z_{ij}** (maximized in "analysis" view):

$$\begin{aligned} \frac{1}{nk} \sum_{i=1}^n \|z_i - \mu_z\|^2 &= \frac{1}{nk} \sum_{i=1}^n \|W x_i\|^2 \quad (\mu_z = 0 \text{ and } z_i = W x_i \text{ if } \|W_c\|=1 \text{ and } W_c^i W_c^i = 0) \\ &= \frac{1}{nk} \sum_{i=1}^n x_i^T W^T W x_i = \frac{1}{nk} \sum_{i=1}^n \text{Tr}(x_i^T W^T W x_i) = \frac{1}{nk} \sum_{i=1}^n \text{Tr}(W^T W x_i x_i^T) \\ &= \frac{1}{nk} \text{Tr}(W^T W \underbrace{\sum_{i=1}^n x_i x_i^T}_{X^T X}) = \frac{1}{nk} \text{Tr}(W^T W X^T X) \end{aligned}$$

linearity of trace (pointing to the sum in the third term)

"cyclic" property of trace (pointing to the transition from the second to the third term)

- The **distance to the hyper-plane** (minimized in "synthesis" view):

$$\begin{aligned} \|ZW - X\|_F^2 &= \|XW^T W - X\|_F^2 = \text{Tr}((XW^T W - X)^T (XW^T W - X)) \\ &= \text{Tr}(W^T W X^T X W^T W) - 2 \text{Tr}(W^T W X^T X) + \text{Tr}(X^T X) \\ &= \text{Tr}(W^T \underbrace{W W^T}_I W X^T X) - 2 \text{Tr}(W^T W X^T X) + \text{Tr}(X^T X) \\ &= -\text{Tr}(W^T W X^T X) + (\text{constant}) \end{aligned}$$

$\|A\|_F^2 = \text{Tr}(A^T A)$ (pointing to the first term)

$= XW^T$ (pointing to the second term)

Solved by same 'W' (pointing to the final result)

Background Subtraction with Robust PCA

- Robust PCA methods use the **absolute error**:

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

- Will be **robust to outliers** in the matrix 'X'.
- Encourages "residuals" r_{ij} to be exactly zero.
 - Non-zero r_{ij} are where the "outliers" are.

x_{ij} $(w_j^i)^T z_i$ r_{ij}

Applying robust PCA
to video frames



Digression: “Whitening”

- With image data, features will be very redundant.
 - Neighbouring pixels tend to have similar values.
- A standard transformation in these settings is “whitening”:
 - Rotate the data so features are uncorrelated.
 - Re-scale the rotated features so they have a variance of 1.
- Using SVD approach to PCA, we can do this with:
 - Get ‘W’ from SVD (usually with $k=d$).
 - $Z = XW^T$ (rotate to give uncorrelated features).
 - Divide columns of ‘Z’ by corresponding singular values (unit variance).
- Details/discussion [here](#).

Kernel PCA

- From the “analysis” view (with orthogonal PCs) PCA maximizes:

$$\text{Tr}(W^T W X^T X)$$

- It can be shown that the solution has the form (see [here](#)):

$$\underbrace{W}_{K \times d} = \underbrace{U}_{K \times n} \underbrace{X}_{n \times 1}$$

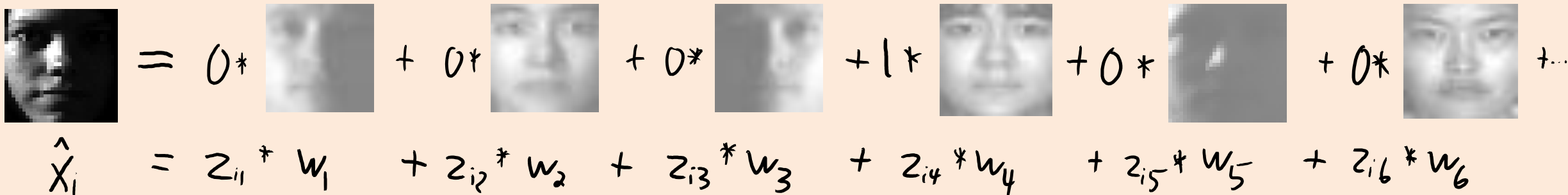
- Re-parameterizing in terms of ‘U’ gives a **kernelized PCA**:

$$\text{Tr}(X^T U^T U X X^T X) = \text{Tr}(U^T U \underbrace{X X^T}_K \underbrace{X X^T}_K)$$

- It’s hard to initially center data in ‘Z’ space, but you can **form the centered kernel matrix** (see [here](#)).

VQ vs. PCA vs. NMF

- How *should* we represent faces?
 - **Vector quantization** (k-means).
 - Replace face by the **average face in a cluster**.
 - ‘Grandmother cell’: one neuron = one face.
 - **Can’t distinguish between people** in the same cluster (only ‘k’ possible faces).
 - Almost certainly not true: too few neurons.


$$\hat{X}_i = z_{i1} * w_1 + z_{i2} * w_2 + z_{i3} * w_3 + z_{i4} * w_4 + z_{i5} * w_5 + z_{i6} * w_6 + \dots$$

VQ vs. PCA vs. NMF

- How *should* we represent faces?
 - **Vector quantization** (k-means).
 - **PCA** (orthogonal basis).
 - Global average plus **linear combination** of “eigenfaces”.
 - “Distributed representation”.
 - Coded by **pattern of group** of neurons: can represent **infinite number of faces** by changing z_i .
 - But “eigenfaces” are **not intuitive** ingredients for faces.
 - PCA tends to use positive/negative **cancelling** bases.

$$\hat{x}_1 = \mu + z_{i1} * w_1 + z_{i2} * w_2 + z_{i3} * w_3 + z_{i4} * w_4 + z_{i5} * w_5 + \dots$$

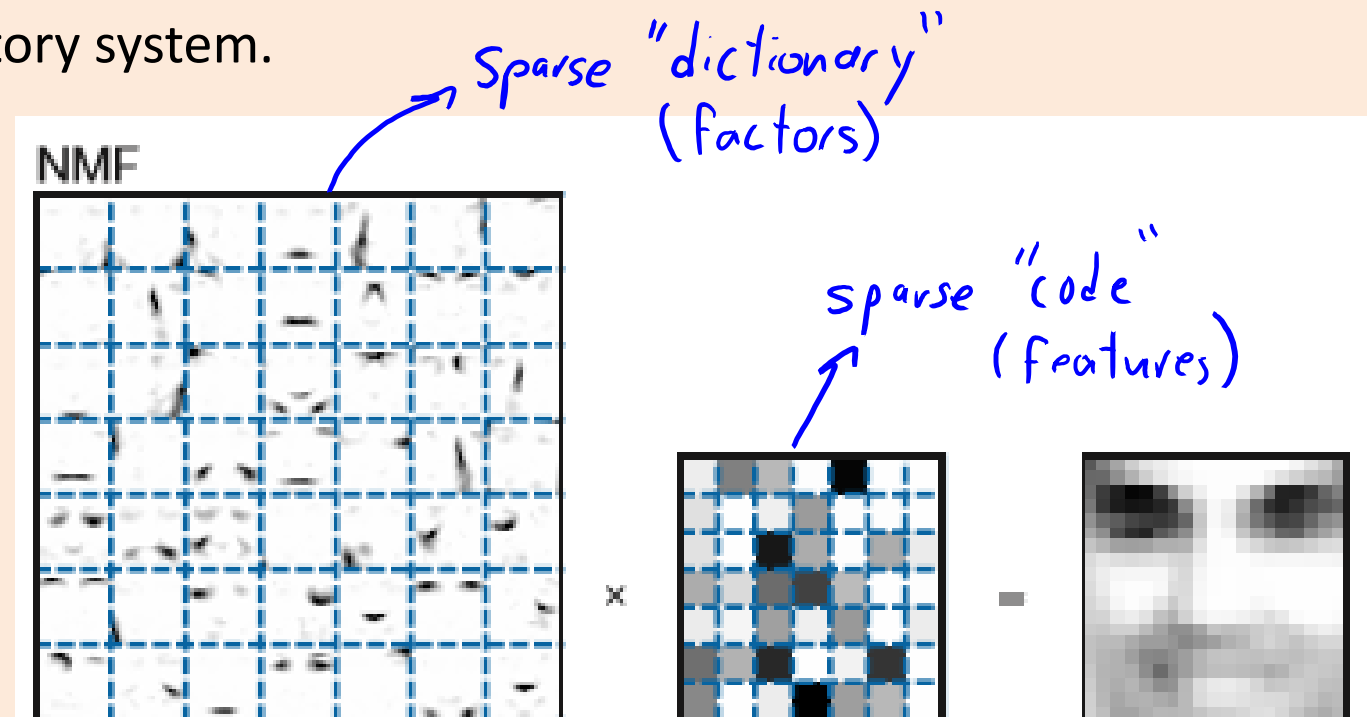
VQ vs. PCA vs. NMF

- How *should* we represent faces?
 - Vector quantization (k-means).
 - PCA (orthogonal basis).
 - NMF (non-negative matrix factorization):
 - Instead of orthogonality/ordering in W , require W and Z to be **non-negativity**.
 - Example of “**sparse coding**”:
 - The z_i are **sparse** so each face is coded by a **small number of neurons**.
 - The w_c are **sparse** so neurons tend to be “**parts**” of the object.

$$\hat{X}_i = z_{i1} * w_1 + z_{i2} * w_2 + z_{i3} * w_3 + z_{i4} * w_4 + z_{i5} * w_5 + \dots$$

Representing Faces

- Why sparse coding?
 - “Parts” are intuitive, and brains seem to use sparse representation.
 - Energy efficiency if using sparse code.
 - Increase number of concepts you can memorize?
 - Some evidence in fruit fly olfactory system.



Warm-up to NMF: Non-Negative Least Squares

- Consider our usual **least squares** problem:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w^T x_i - y_i)^2$$

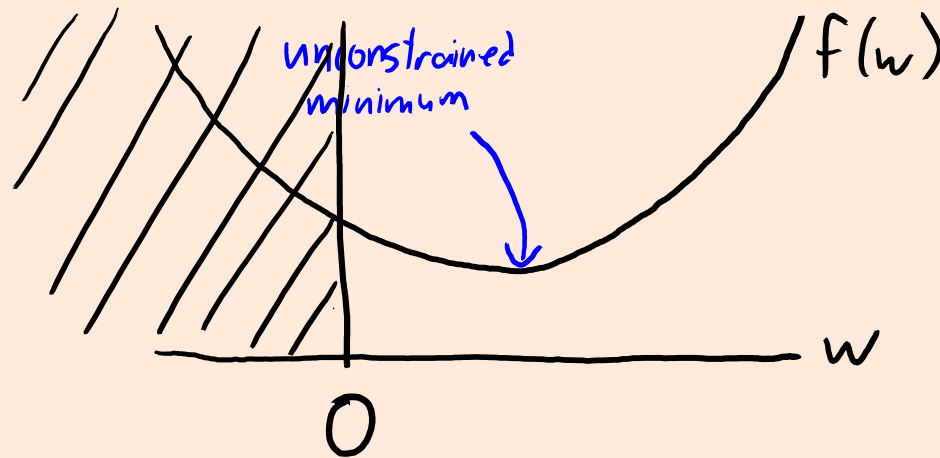
- But assume **y_i and elements of x_i are non-negative**:
 - Could be sizes ('height', 'milk', 'km') or counts ('vicodin', 'likes', 'retweets').
- Assume we want elements of ' **w** ' to be **non-negative**, too:
 - **No physical interpretation to negative weights.**
 - If x_{ij} is amount of product you produce, what does $w_j < 0$ mean?
- **Non-negativity leads to sparsity...**

Sparsity and Non-Negative Least Squares

- Consider 1D non-negative least squares objective:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w x_i - y_i)^2 \quad \text{with } w \geq 0$$

- Plotting the (constrained) objective function:



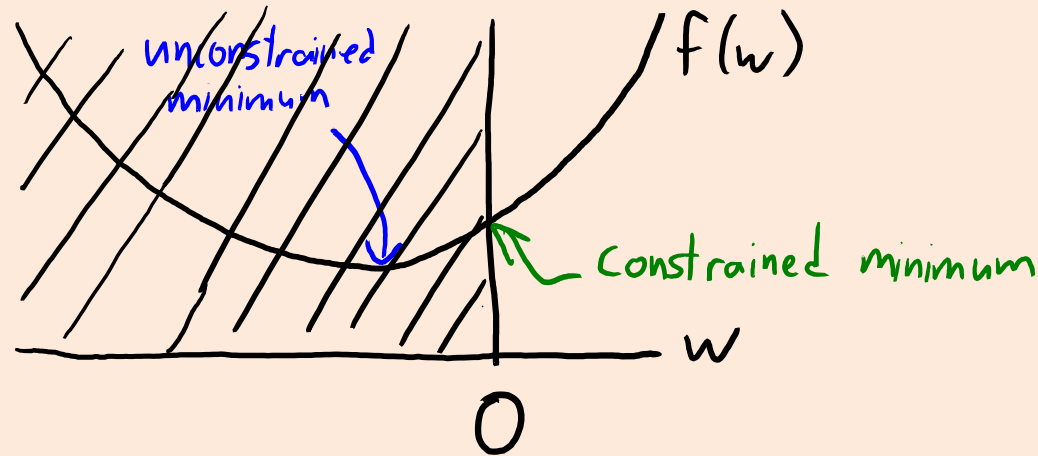
- In this case, non-negative solution is least squares solution.

Sparsity and Non-Negative Least Squares

- Consider 1D non-negative least squares objective:

$$f(w) = \frac{1}{2} \sum_{i=1}^n (w x_i - y_i)^2 \quad \text{with } w \geq 0$$

- Plotting the (constrained) objective function:



- In this case, **non-negative solution is $w = 0$.**

Sparsity and Non-Negativity

- Similar to L1-regularization, **non-negativity leads to sparsity**.
 - Also **regularizes**: w_j are smaller since can't "cancel" negative values.
 - Sparsity leads to **cheaper predictions** and often to more interpretability.
 - Non-negative weights are often also **more interpretable**.

- How can we minimize $f(w)$ with **non-negative constraints**?
 - **Naive approach**: solve least squares problem, set negative w_j to 0.

$$\text{Compute } w = (X^T X)^{-1} X^T y$$

$$\text{Set } w_j = \max\{0, w_j\}$$

- This is correct when $d = 1$.
- **Can be worse than setting $w = 0$** when $d \geq 2$.

Sparsity and Non-Negativity

- Similar to L1-regularization, **non-negativity leads to sparsity**.
 - Also **regularizes**: w_j are smaller since can't “cancel” out negative values.
- How can we minimize $f(w)$ with **non-negative constraints**?
 - A correct approach is **projected gradient** algorithm:
 - Run a **gradient descent** iteration:

$$w^{t+1/2} = w^t - \alpha^t \nabla f(w^t)$$

- **After each step, set negative values to 0.**

$$w_j^{t+1} = \max\{0, w_j^{t+1/2}\}$$

- Repeat.

Sparsity and Non-Negativity

- Projected gradient algorithm:

$$w^{t+1/2} = w^t - \alpha^t \nabla f(w^t)$$

$$w_j^{t+1} = \max\{0, w_j^{t+1/2}\}$$

- Similar properties to gradient descent:

- Guaranteed decrease of 'f' if α^t is small enough.
- Reaches local minimum under weak assumptions (global minimum for convex 'f').
 - Least squares objective is still convex when restricted to non-negative variables.
- Solution is a “fixed point”: $w^* = \max\{0, w^* - \alpha^t \nabla f(w^*)\}$.
 - Use this to decide when to stop.

- A generalization is “proximal-gradient”:

- Instead of constraints, allows non-smooth terms (“findMinL1”).

Projected-Gradient for NMF

- Back to the **non-negative matrix factorization (NMF)** objective:

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d (\langle w_j, z_i \rangle - x_{ij})^2 \quad \text{with } w_{cj} \geq 0 \text{ and } z_{ij} \geq 0$$

– Different ways to use **projected gradient**:

- Alternate between projected gradient steps on 'W' and on 'Z'.
- Or run projected gradient on both at once.
- Or sample a random 'i' and 'j' and do **stochastic projected gradient**.

Set $z_i^{t+1} = z_i^t - \alpha^t \nabla_{z_i} f(W, Z)$ and $(w_j)^{t+1} = (w_j)^t - \alpha^t \nabla_{w_j} f(W, Z)$ for selected i and j

– **Non-convex** and (unlike PCA) is **sensitive to initialization**.

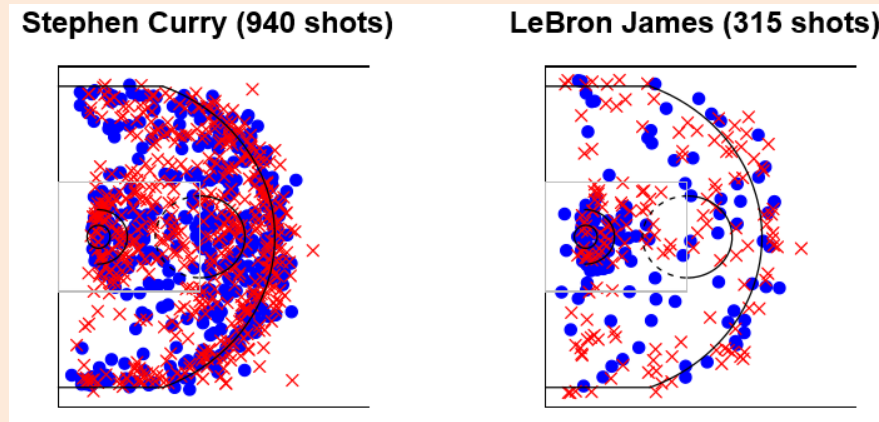
- Hard to find the global optimum.
- Typically use **random initialization**.
- Also, we **usually don't center the data** with NMF.

(keep other values of W and Z fixed)

Then set negative values to 0.

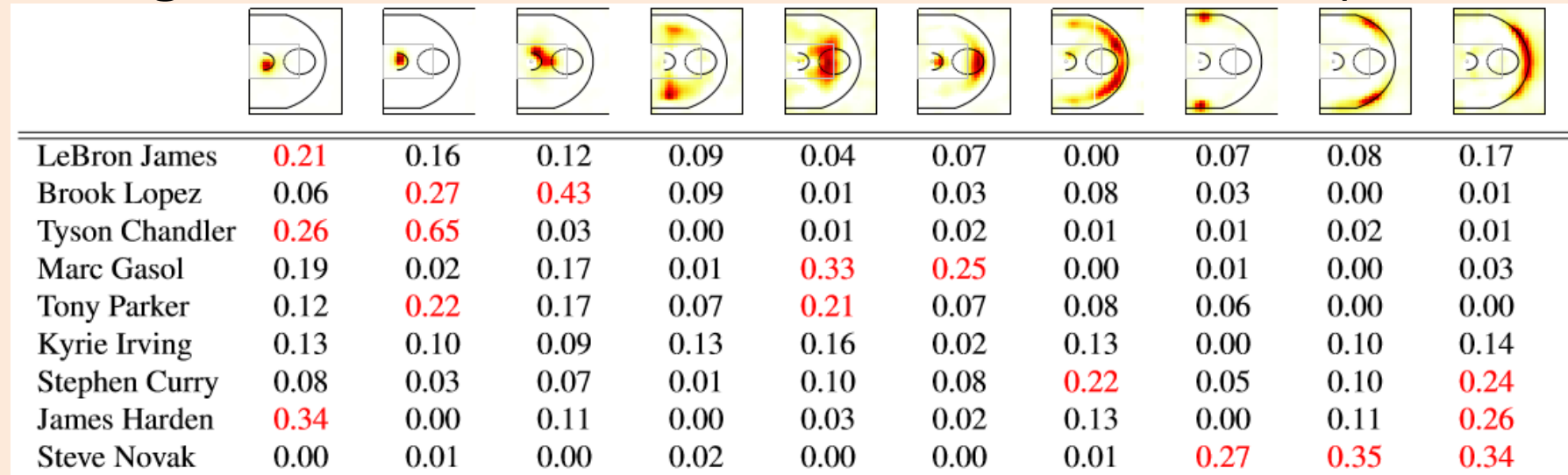
Application: Sports Analytics

- NBA shot charts:



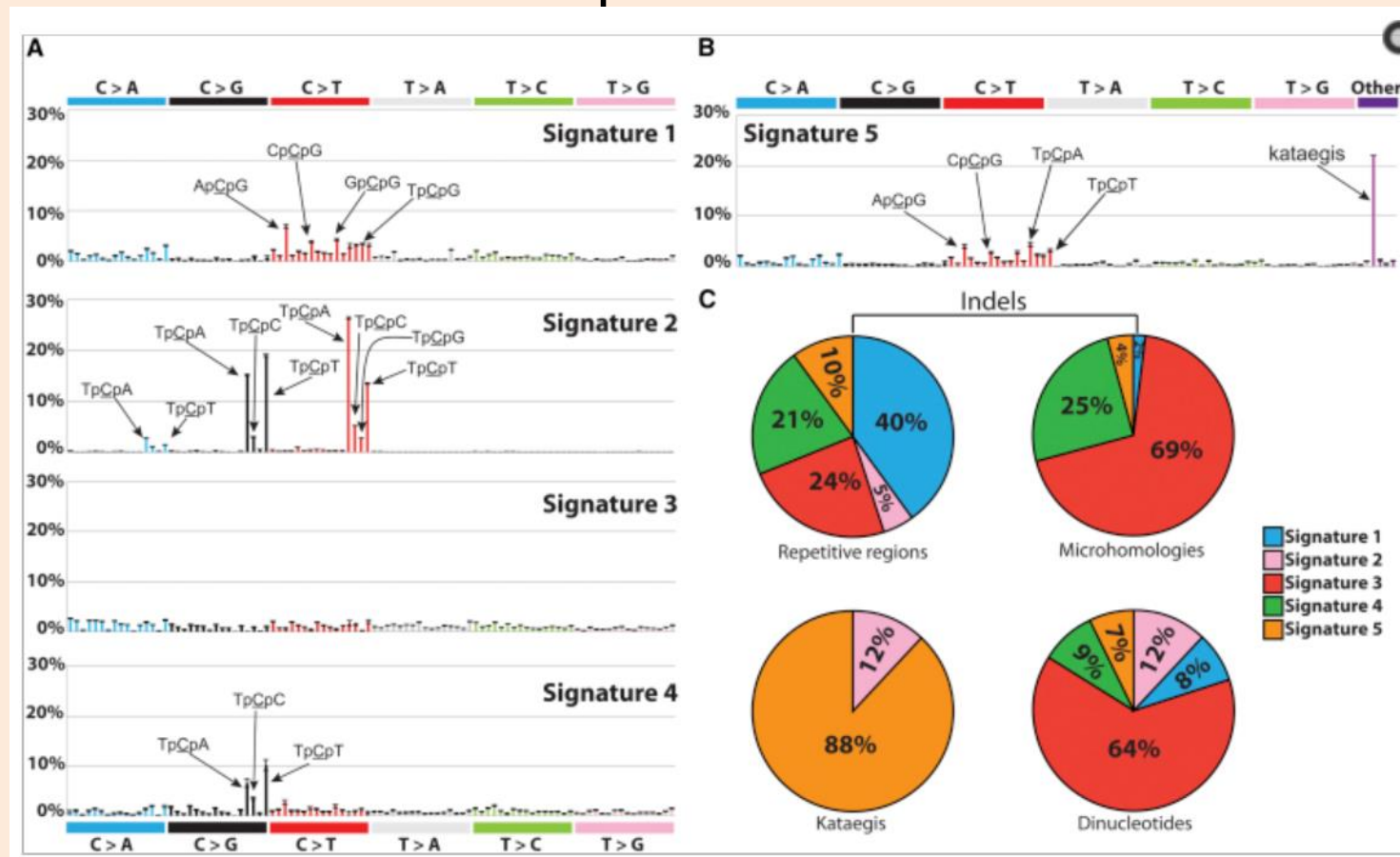
- NMF (using “KL divergence” loss with $k=10$ and smoothed data).

– Negative values would not make sense here.



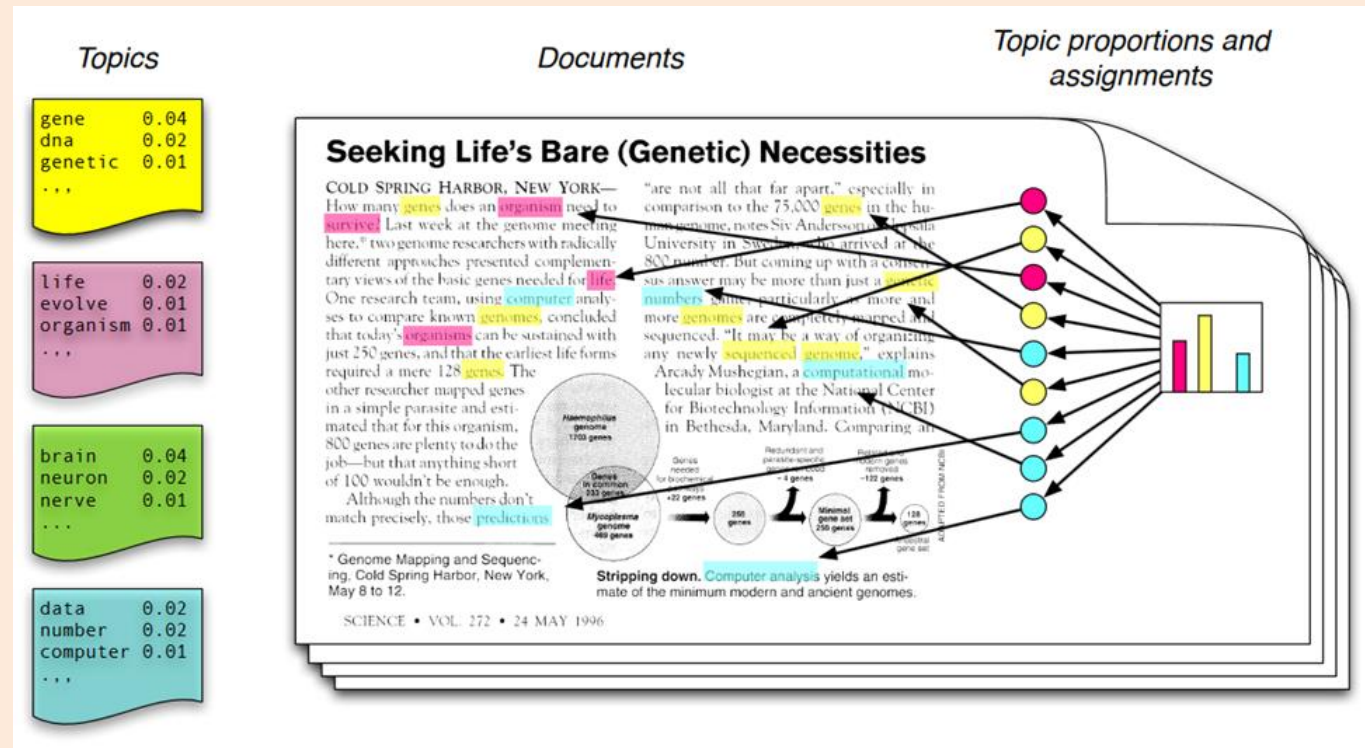
Application: Cancer “Signatures”

- What are common sets of mutations in different cancers?
 - May lead to new treatment options.



Beyond Matrix Factorization: Topic Models

- For modeling categorical data, “topic models” are replacing NMF.
 - A “fully-Bayesian” model where sparsity arises naturally.
 - Most popular example is called “latent Dirichlet allocation” (CPSC 440).



Sparse Matrix Factorization

- Instead of non-negativity, we could use L1-regularization:

$$f(W, Z) = \frac{1}{2} \|ZW - X\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^n \|z_i\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^d \|w_j\|_1$$

- Called **sparse coding** (L1 on 'Z') or **sparse dictionary learning** (L1 on 'W').
- **Disadvantage of using L1-regularization** over non-negativity:
 - Sparsity controlled by λ_1 and λ_2 so you need to set these.
- **Advantage of using L1-regularization:**
 - Sparsity controlled by λ_1 and λ_2 , so you can **control amount of sparsity**.
 - Negative coefficients often do make sense.

Sparse Matrix Factorization

- Instead of non-negativity, we could use L1-regularization:

$$f(W, Z) = \frac{1}{2} \|ZW - X\|_F^2 + \frac{\lambda_1}{2} \sum_{i=1}^n \|z_i\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^d \|w_j\|_1$$

- Called **sparse coding** (L1 on 'Z') or **sparse dictionary learning** (L1 on 'W').
- Many variations exist:
 - Mixing L2-regularization and L1-regularization.
 - Or normalizing 'W' (in L2-norm or L1-norm) and regularizing 'Z'.
 - **K-SVD** constrains each z_i to have at most 'k' non-zeroes:
 - K-means is special case where $k = 1$.
 - PCA is special case where $k = d$.

Canonical Correlation Analysis (CCA)

- Suppose we have two matrices, 'X' and 'Y'.
- Want to find matrices W_X and W_Y that maximize correlation.
 - “What are the latent factors in common between these datasets?”
- Define the correlation matrices:

$$\Sigma_{XX} = \frac{1}{n} \sum_{i=1}^n x_i x_i^T \quad \Sigma_{YY} = \frac{1}{n} \sum_{i=1}^n y_i y_i^T \quad \Sigma_{XY} = \frac{1}{n} \sum_{i=1}^n x_i y_i^T$$

- **Canonical correlation analysis (CCA)** maximizes

$$\text{Tr}(W_Y^T W_X \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2})$$

- Subject to W_X and W_Y having orthogonal rows.
- Computationally, equivalent to **PCA with a different matrix**.
 - Using the “analysis” view that PCA maximizes $\text{Tr}(W^T W X^T X)$.

Probabilistic PCA

- With zero-mean (“centered”) data, in PCA we assume that

$$x_i \approx W^T z_i$$

- In **probabilistic PCA** we assume that

$$x_i \sim \mathcal{N}(W^T z_i, \sigma^2 I) \quad z_i \sim \mathcal{N}(0, I)$$

- Integrating over ‘Z’ the marginal likelihood given ‘W’ is Gaussian,

$$x_i | W \sim \mathcal{N}(0, W^T W + \sigma^2 I)$$

- Regular PCA is obtained as the limit of σ^2 going to 0.

Generalizations of Probabilistic PCA

- Probabilistic PCA model:

$$x_i | W \sim \mathcal{N}(0, W^T W + \sigma^2 I)$$

- Why do we need a probabilistic interpretation?
- Shows that **PCA fits a Gaussian with restricted covariance.**
 - Hope is that $W^T W + \sigma^2 I$ is a good approximation of $X^T X$.
- Gives precise connection between PCA and **factor analysis.**

Factor Analysis

- Factor analysis is a method for discovering latent factors.
- Historical applications are measures of intelligence and personality.

Trait	Description
O penness	Being curious, original, intellectual, creative, and open to new ideas.
C onscientiousness	Being organized, systematic, punctual, achievement-oriented, and dependable.
E xtraversion	Being outgoing, talkative, sociable, and enjoying social situations.
A greeableness	Being affable, tolerant, sensitive, trusting, kind, and warm.
N euroticism	Being anxious, irritable, temperamental, and moody.

- A standard tool and widely-used across science and engineering.

PCA vs. Factor Analysis

- PCA and FA both write the matrix 'X' as

$$X \approx ZW$$

- PCA and FA are both based on a Gaussian assumption.
- Are PCA and FA the same?
 - Both are more than 100 years old.
 - People are still arguing about whether they are the same:
 - Doesn't help that some packages run PCA when you call their FA method.



All Images Videos News Maps More Search tools

About 358,000 results (0.17 seconds)

[PDF] Principal Component Analysis versus Exploratory Factor ...

www2.sas.com/proceedings/sugi30/203-30.pdf

by DD Suhr - Cited by 118 - Related articles

1. Paper 203-30. Principal Component Analysis vs. Exploratory Factor Analysis.
Diana D. Suhr, Ph.D. University of Northern Colorado. Abstract. Principal ...

pca - What are the differences between Factor Analysis and ...

stats.stackexchange.com/.../what-are-the-differences-between-factor-anal...

Aug 12, 2010 - Principal Component Analysis (PCA) and Common Factor Analysis (CFA) differently one has to interpret the strength of loadings in PCA vs.

What are the differences between principal components ...

support.minitab.com/.../factor-analysis/differences-between-pca-and-facto...

Principal Components Analysis and Factor Analysis are similar because both procedures are used to simplify the structure of a set of variables. However, the ...

[PDF] Principal Components Analysis - UNT

<https://www.unt.edu/rss/class/.../Principal%20Components%20Analysis.p...>

PCA vs. Factor Analysis. • It is easy to make the mistake in assuming that these are the same techniques, though in some ways exploratory factor analysis and ...

Factor analysis versus Principal Components Analysis (PCA)

psych.wisc.edu/henriques/pca.html

Jun 19, 2010 - Factor analysis versus PCA. These techniques are typically used to analyze groups of correlated variables representing one or more common ...

[PDF] Principal Component Analysis and Factor Analysis

www.stats.ox.ac.uk/~ripley/MultAnal_HT2007/PC-FA.pdf

where D is diagonal with non-negative and decreasing values and U and V
Factor analysis and PCA are often confused, and indeed SPSS has PCA as.

How can I decide between using principal components ...

https://www.researchgate.net/.../How_can_I_decide_between_using_prin...

Factor analysis (FA) is a group of statistical methods used to understand and simplify patterns ... Retrieved from <http://pareonline.net/getvn.asp?v=10&n=7> ...
Principal component analysis (PCA) is a method of factor extraction (the second step ...

[PDF] Exploratory Factor Analysis and Principal Component An...

www.lesahoffman.com/948/948_Lecture2_EFA_PCA.pdf

2 very different schools of thought on exploratory factor analysis (EFA) vs. principal components analysis (PCA): > EFA and PCA are TWO ENTIRELY ...

Factor analysis - Wikipedia, the free encyclopedia

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

Jump to **Exploratory factor analysis versus principal components ...** - [edit]. See also: Principal component analysis and Exploratory factor analysis.

[PDF] The Truth about PCA and Factor Analysis

www.stat.cmu.edu/~cshalizi/350/lectures/13/lecture-13.pdf

Sep 28, 2009 - nents and factor analysis, we'll wrap up by looking at their uses and

PCA vs. Factor Analysis

- In probabilistic PCA we assume:

$$x_i \sim \mathcal{N}(W^T z_i, \sigma^2 I)$$

- In FA we assume for a diagonal matrix **D** that:

$$x_i \sim \mathcal{N}(W^T z_i, D)$$

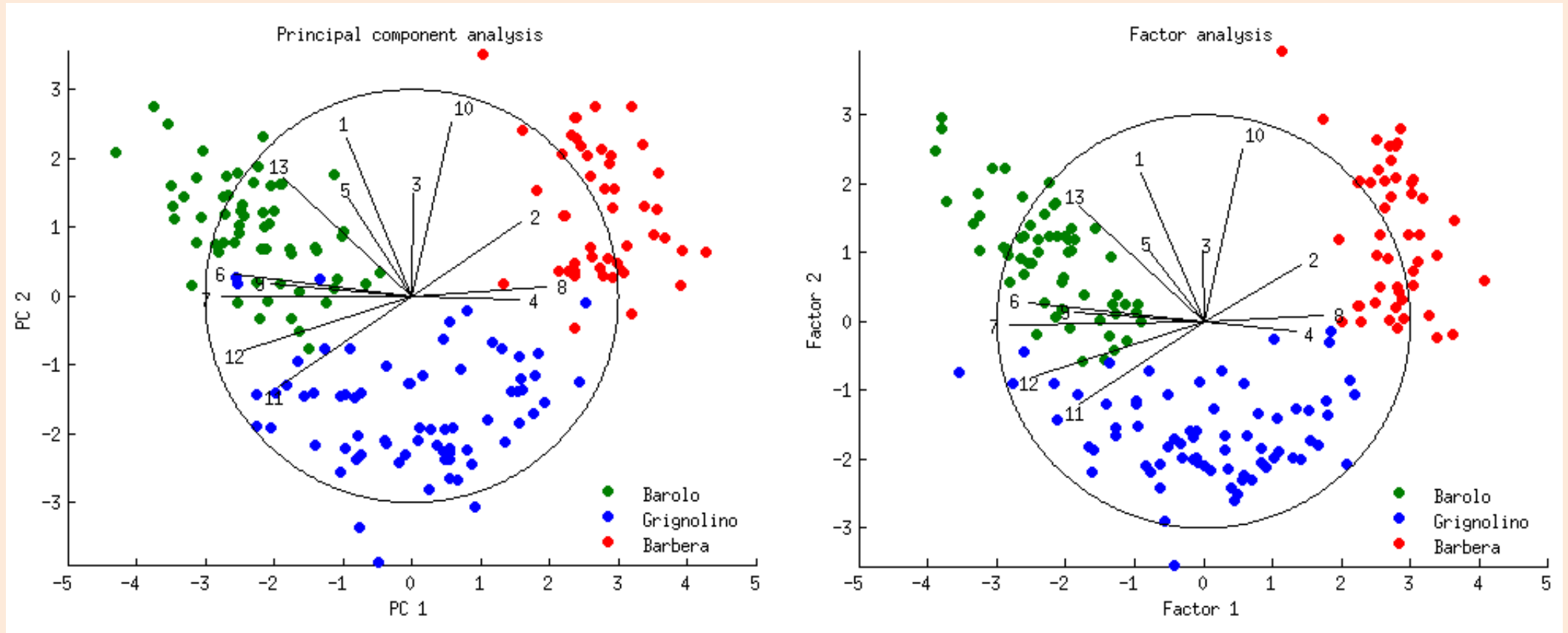
- The posterior in this case is: $x_i | W \sim \mathcal{N}(0, W^T W + D)$

- The difference is you have a **noise variance for each dimension.**
 - FA has extra degrees of freedom.



PCA vs. Factor Analysis

- In practice there often isn't a huge difference:



Factor Analysis Discussion

- Differences with PCA:
 - Unlike PCA, FA is not affected by scaling individual features.
 - But unlike PCA, it's affected by rotation of the data.
 - No nice “SVD” approach for FA, you can get different local optima.
- Similar to PCA, FA is invariant to rotation of ‘W’.
 - So as with PCA you can't interpret multiple factors as being unique.

Motivation for ICA

- Factor analysis has found an enormous number of applications.
 - People really want to find the “hidden factors” that make up their data.
- But PCA and FA **can't identify the factors.**

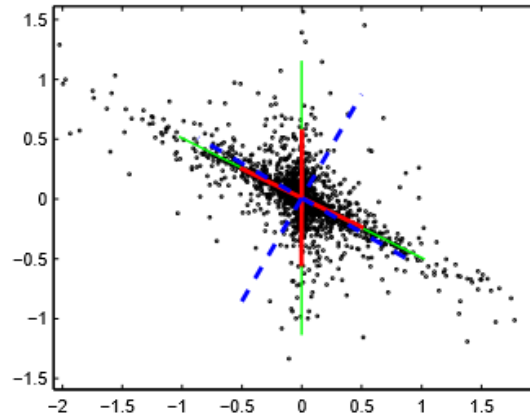


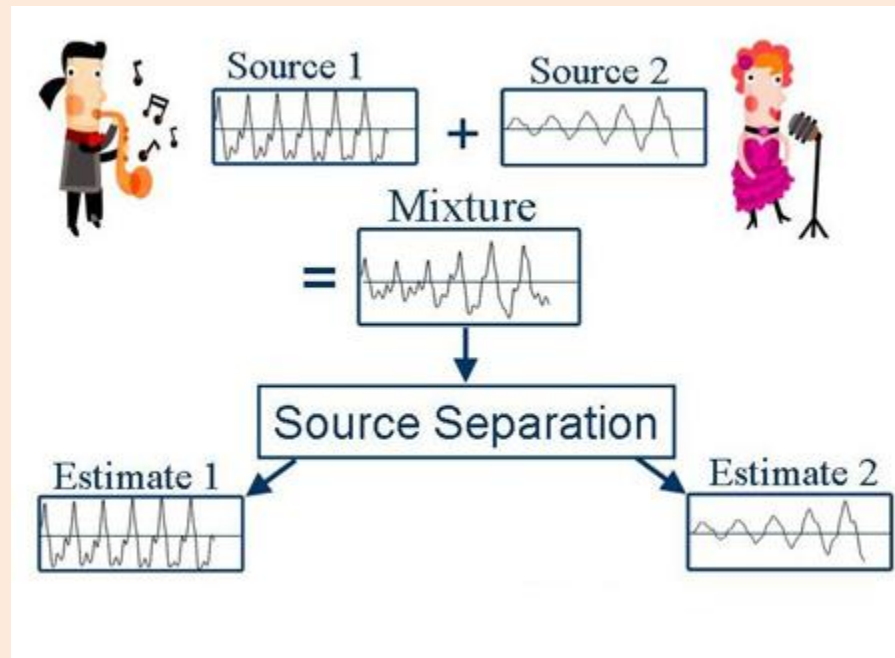
Figure : Latent data is sampled from the prior $p(x_i) \propto \exp(-5\sqrt{|x_i|})$ with the mixing matrix A shown in green to create the observed two dimensional vectors $y = Ax$. The red lines are the mixing matrix estimated by `ica.m` based on the observations. For comparison, PCA produces the blue (dashed) components. Note that the components have been scaled to improve visualisation. As expected, PCA finds the orthogonal directions of maximal variation. ICA however, correctly estimates the directions in which the components were independently generated.

Motivation for ICA

- Factor analysis has found an enormous number of applications.
 - People really want to find the “hidden factors” that make up their data.
- But PCA and FA **can't identify the factors**.
 - We can rotate W and obtain the same model.
- **Independent component analysis (ICA)** is a more recent approach.
 - Around 30 years old instead of > 100 .
 - Under certain assumptions it can **identify factors**.
- The canonical application of ICA is **blind source separation**.

Blind Source Separation

- Input to **blind source separation**:
 - **Multiple microphones** recording **multiple sources**.



- Each microphone gets different mixture of the sources.
 - Goal is reconstruct sources (factors) from the measurements.

Independent Component Analysis Applications

- ICA is replacing PCA and FA in many applications:

Some ICA applications are listed below:^[1]

- optical Imaging of neurons^[17]
- neuronal spike sorting^[18]
- face recognition^[19]
- modeling receptive fields of primary visual neurons^[20]
- predicting stock market prices^[21]
- mobile phone communications ^[22]
- color based detection of the ripeness of tomatoes^[23]
- removing artifacts, such as eye blinks, from EEG data.^[24]

- Recent work shows that ICA can often resolve **direction of causality**.

Limitations of Matrix Factorization

- ICA is a **matrix factorization** method like PCA/FA,

$$X = ZW$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.
 - Different from PCA where we assume $k \leq d$.
- There are only **3 issues stopping us from finding "true" W** .

3 Sources of Matrix Factorization Non-Uniqueness

- **Label switching**: get same model if we **permute rows** of W .
 - We can exchange row 1 and 2 of W (and same columns of Z).
 - Not a problem because we don't care about order of factors.
- **Scaling**: get same model if you **scale a row**.
 - If we multiply row 1 of W by α , could multiply column 1 of Z by $1/\alpha$.
 - Can't identify sign/scale, but might hope to identify direction.
- **Rotation**: get same model if we **rotate W** .
 - Rotations correspond to orthogonal matrices Q , such matrices have $Q^T Q = I$.
 - If we rotate W with Q , then we have $(QW)^T QW = W^T Q^T QW = W^T W$.
- **If we could address rotation, we could identify the “true” directions.**

A Unique Gaussian Property

- Consider an **independent prior on each latent features z_c** .
 - E.g., in PPCA and FA we use $N(0,1)$ for each z_c .
- If prior $p(z)$ is independent and **rotation-invariant** ($p(Qz) = p(z)$), then it must be Gaussian (only Gaussians have this property).
- The (non-intuitive) magic behind ICA:
 - If the priors are all **non-Gaussian**, it **isn't rotationally symmetric**.
 - In this case, we can **identify factors W** (up to permutations and scalings).

PCA vs. ICA

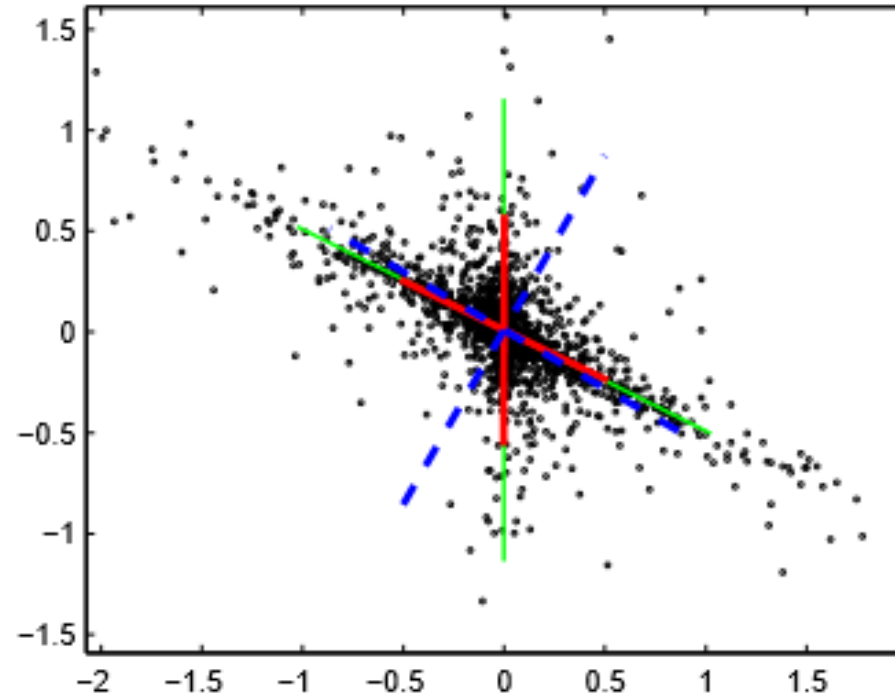


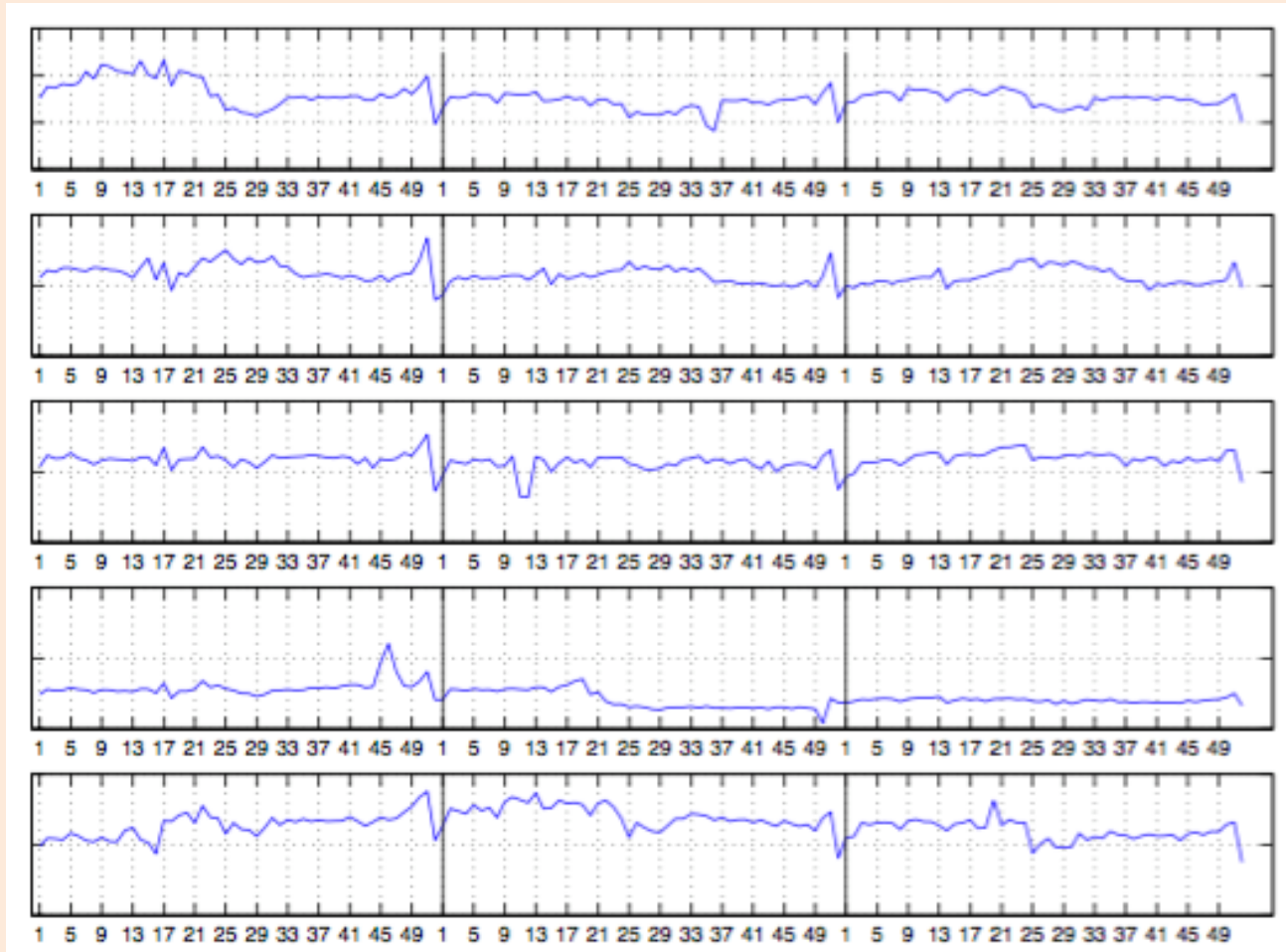
Figure : Latent data is sampled from the prior $p(x_i) \propto \exp(-5 \sqrt{|x_i|})$ with the mixing matrix A shown in green to create the observed two dimensional vectors $y = Ax$. The red lines are the mixing matrix estimated by `ica.m` based on the observations. For comparison, PCA produces the blue (dashed) components. Note that the components have been scaled to improve visualisation. As expected, PCA finds the orthogonal directions of maximal variation. ICA however, correctly estimates the directions in which the components were independently generated.

Independent Component Analysis

- In ICA we approximate X with ZW , assuming $p(z_{ic})$ are **non-Gaussian**.
- Usually we “center” and “whiten” the data before applying ICA.
- There are several penalties that encourage non-Gaussianity:
 - Penalize low **kurtosis**, since kurtosis is minimized by Gaussians.
 - Penalize high **entropy**, since entropy is maximized by Gaussians.
- The **fastICA** is a popular method maximizing kurtosis.

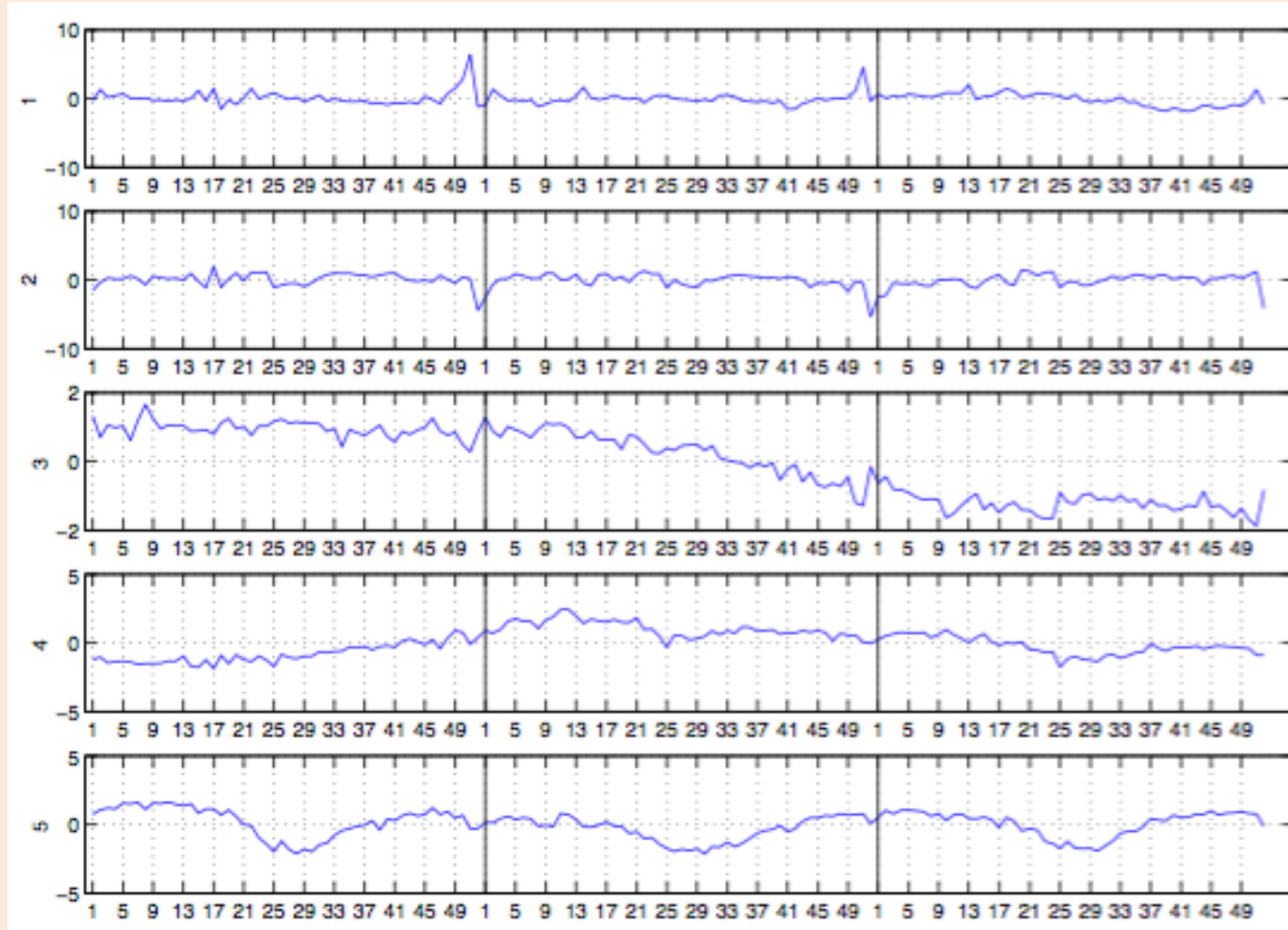
ICA on Retail Purchase Data

- Cash flow from 5 stores over 3 years:



ICA on Retail Purchase Data

- Factors found using ICA:



Motivation for Topic Models

- Want a model of the “factors” making up documents.
 - Instead of latent-factor models, they’re called **topic models**.
 - The canonical topic model is **latent Dirichlet allocation (LDA)**.

Suppose you have the following set of sentences:

- I like to eat broccoli and bananas.
- I ate a banana and spinach smoothie for breakfast.
- Chinchillas and kittens are cute.
- My sister adopted a kitten yesterday.
- Look at this cute hamster munching on a piece of broccoli.

What is latent Dirichlet allocation? It’s a way of automatically discovering **topics** that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like

- **Sentences 1 and 2:** 100% Topic A
- **Sentences 3 and 4:** 100% Topic B
- **Sentence 5:** 60% Topic A, 40% Topic B
- **Topic A:** 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ... (at which point, you could interpret topic A to be about food)
- **Topic B:** 20% chinchillas, 20% kittens, 20% cute, 15% hamster, ... (at which point, you could interpret topic B to be about cute animals)

- “Topics” could be useful for things like searching for relevant documents.

Term Frequency – Inverse Document Frequency

- In information retrieval, classic word importance measure is **TF-IDF**.
- First part is the **term frequency** $tf(t,d)$ of term 't' for document 'd'.
 - Number of times “word” ‘t’ occurs in document ‘d’, divided by total words.
 - E.g., 7% of words in document ‘d’ are “the” and 2% of the words are “Lebron”.
- Second part is **document frequency** $df(t,D)$.
 - Compute **number of documents that have ‘t’** at least once.
 - E.g., 100% of documents contain “the” and 0.01% have “LeBron”.
- TF-IDF is $tf(t,d) * \log(1/df(t,D))$.

Term Frequency – Inverse Document Frequency

- The **TF-IDF** statistic is $tf(t,d) * \log(1/df(t,D))$.
 - It's high if word 't' happens often in document 'd', but isn't common.
 - E.g., seeing "LeBron" a lot it tells you something about "topic" of article.
 - E.g., seeing "the" a lot tells you nothing.
- There are **many** variations on this statistic.
 - E.g., avoiding dividing by zero and all types of "frequencies".
- Summarizing 'n' documents into a matrix X:
 - Each row corresponds to a document.
 - Each column gives the TF-IDF value of a particular word in the document.

Latent Semantic Indexing

- TF-IDF features are **very redundant**.
 - Consider TF-IDFs of “LeBron”, “Durant”, “Harden”, and “Kobe”.
 - High values of these typically just indicate topic of “basketball”.
- We can probably compress this information quite a bit.
- Latent Semantic Indexing/Analysis:
 - Run **latent-factor model (like PCA or NMF)** on TF-IDF matrix X .
 - Treat the principal components as the “topics”.
 - **Latent Dirichlet allocation** is a variant that avoids weird $df(t,D)$ heuristic.