CPSC 340:
Machine Learning and Data Mining

Sparse Matrix Factorization
Fall 2019
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** \( \mathbf{W}_c \) of ‘\( \mathbf{W} \)’.
    \[
    \| \mathbf{w}_c \| = 1 \quad \text{and} \quad \mathbf{w}_c^T \mathbf{w}_{c'} = 0 \quad \text{for} \quad c' \neq c
    \]
  – And **fit the rows in order**:
    • First row “explains the most variance” or “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.

http://oneminuteastronomer.com/astro-course-day-5/
https://en.wikipedia.org/wiki/Color_vision
• 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

\[ w_{1} \]

First row of \( W \) (First PC)

\( +w_{2} \)

Second row \( (4 \times 1) \)

brightness

\( +w_{3} \)

Third row \( (4 \times 1) \)

red/green

Analogous to means in k-means.

Can represent 4 original values with these 3 \( 2 \times 1 \) values and matrix 'W'

blue/yellow

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:

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

• 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:

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

• 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:

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

• **Alternating minimization** steps:
  
  – If we fix \( Z \), this is a quadratic function of \( W \) (least squares column-wise):
    
    \[ \nabla_w f(w, z) = Z^T Z W - Z^T X \quad \text{so} \quad W = (Z^T Z)^{-1}(Z^T X) \]
    
    (writing gradient as a matrix)

  – If we fix \( W \), this is a quadratic function of \( Z \) (transpose due to dimensions):
    
    \[ \nabla_z f(w, z) = Z W W^T - X W^T \quad \text{so} \quad Z = X W^T (WW^T)^{-1} \]
PCA Computation: Alternating Minimization

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

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

• 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 \(W_1 = 0\), 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

• For big X matrices, you can also use stochastic gradient:

\[
f(W, z) = \sum_{i=1}^{d} \sum_{j=1}^{d} (\langle w_j^iz_i \rangle - x_{ij})^2 = \sum_{(i,j)} (\langle w_j^iz_i \rangle - x_{ij})^2 = f(w_j^iz_i, x_{ij})
\]

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

\[\rightarrow \text{Set } w_j^i \text{ to } w_j^i - \alpha^t \nabla_w f(w_j^i, z_i, x_{ij})\]

\[\rightarrow \text{Set } z_i \text{ to } z_i - \alpha^t \nabla_{z_i} f(w_j^i, z_i, x_{ij})\]

• Other variables stay the same, cost per iteration is only O(k).
(pause)
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.

\[
\hat{X}_1 = z_{i1} \cdot w_1 + z_{i2} \cdot w_2 + z_{i3} \cdot w_3 + z_{i4} \cdot w_4 + z_{i5} \cdot w_5 + z_{i6} \cdot w_6 + ... 
\]
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.
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.
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:

\[
\min_w J(w) = \frac{1}{2} \sum_{i=1}^{n} (w^T x_i - y_i)^2
\]

• 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:

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

• Plotting the (constrained) objective function:

• In this case, non-negative solution is least squares solution.
Consider 1D non-negative least squares objective:

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

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:
      \[
      w^{t+\frac{1}{2}} = w^t - \alpha^t \nabla f(w^t)
      \]
    • After each step, set negative values to 0.
      \[
      w_j^{t+1} = \max \{ 0, w_j^{t+\frac{1}{2}} \}
      \]
    • Repeat.
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:

  \[
  w^{t+1} = w^t - \alpha^t \nabla f(w^t) \\
  w_j^{t+1} = \max \{ 0, w_j^{t+1/2} \}
  \]

  – Similar properties to gradient descent:
    • Guaranteed decrease of ‘f’ if $\alpha^t$ 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.
    • Generalizations allow things like L1-regularization instead of non-negativity.

(“findMinL1”)
Projected-Gradient for NMF

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

\[ f(W, Z) = \sum_{i=1}^{n} \sum_{j=1}^{d} (\langle w_i z_j \rangle - x_{ij})^2 \quad \text{with } w_{cj} > 0 \]

and \( z_{ij} > 0 \)

– 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.

\[ s_{t+1} = s_t - \alpha \nabla_{s_t} f(W, Z) \quad \text{and} \quad (w)_{t+1} = (w)^t - \alpha \nabla_{w} f(W, Z) \quad \text{for selected } i \text{ and } j \]

– Non-convex and (unlike PCA) is sensitive to initialization.
  • Hard to find the global optimum.
  • Typically use random initialization.
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.

[Link to NCBI article](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3588146/)
(pause)
Beyond Squared Error

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

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

• As before, there are alternatives to squared error.

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

• If \( X \) consists of +1 and -1 values, we could use logistic loss:

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

- Robust PCA methods use the absolute error:

\[
\min_{(W,z)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} |x_{ij} - \langle w_j, z_i \rangle| 
\]

- 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.

[Images of video frames showing the application of robust PCA to video frames]

http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf
Robust PCA

• Miss Korea contestants and robust PCA:
Regularized Matrix Factorization

• For many PCA applications, ordering orthogonal PCs makes sense.
  – Latent factors are independent of each other.
  – We definitely want this for visualization.

• In other cases, ordering orthogonal PCs doesn’t make sense.
  – We might not expect a natural “ordering”.

Regularized Matrix Factorization

• More recently people have considered \( \text{L2-regularized PCA} \):

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

• Replaces normalization/orthogonality/sequential-fitting.
  – But requires regularization parameters \( \lambda_1 \) and \( \lambda_2 \).

• Need to regularize \( W \) and \( Z \) because of scaling problem.
  – If you only regularize ‘\( W \)’ it doesn’t do anything.
    • I could take unregularized solution, replace \( W \) by \( \alpha W \) for a tiny \( \alpha \) 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 doesn’t do anything.
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 \( \lambda_1 \) and \( \lambda_2 \) so you need to set these.

• Advantage of using L1-regularization:
  - Sparsity controlled by \( \lambda_1 \) and \( \lambda_2 \), 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:

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

– 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 \( k = d \).
Matrix Factorization with L1-Regularization

(a) PCA
(b) 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.

Variations on Latent-Factor Models

• We can use all our tricks for linear regression in this context:

\[ f(W, Z) = \sum_{i=1}^{n} \sum_{j=1}^{d} |<w_j, z_i> - x_{ij}| + \frac{\lambda_1}{2} \sum_{i=1}^{n} \sum_{c=1}^{k} z_{i,c}^2 + \frac{\lambda_2}{2} \sum_{j=1}^{d} \sum_{c=1}^{k} |w_{cj}| \]

• Absolute loss gives robust PCA that is less sensitive to outliers.

• We can use \textbf{L2-regularization}.
  – Though only reduces overfitting if we regularize both ‘W’ and ‘Z’.

• We can use \textbf{L1-regularization} to give sparse latent factors/features.

• We can use logistic/softmax/Poisson losses for discrete \( x_{ij} \).

• Can use change of basis to learn \textbf{non-linear} latent-factor models.
Beyond NMF: Topic Models

- For modeling data as combinations of non-negative parts, NMF has largely replaced by “topic models”.
  - A “fully-Bayesian” model where sparsity arises naturally.
  - Most popular example is called “latent Dirichlet allocation” (CPSC 540).

Summary

• Biological motivation for orthogonal and/or sparse latent factors.
• Alternating minimization and stochastic gradient:
  – Iterative algorithms for minimizing PCA objective.
• Non-negative matrix factorization leads to sparse LFM.
• Non-negativity constraints lead to sparse solution.
  – Projected gradient adds constraints to gradient descent.
  – Non-orthogonal LFM makes sense in many applications.
• Robust PCA allows identifying certain types of outliers.
• L1-regularization leads to other sparse LFM.

• Next time: the million-dollar NetFlix challenge.
Proof: “Synthesis” View = “Analysis” View \((WW^\top = I)\)

- The variance of the \(z_{ij}\) (maximized in “analysis” view):
  
  \[
  \frac{1}{n^k} \sum_{i=1}^{n^k} \|z_i - \mu_2\|^2 = \frac{1}{n^k} \sum_{i=1}^{n^k} \|W_i\|^2 \quad (\mu_2 = 0 \text{ and } z_i = W_i \text{ if } \|W_i\| = 1 \text{ and } W_i^\top W_i = 0)
  \]
  
  \[
  = \frac{1}{n^k} \sum_{i=1}^{n^k} x_i^\top W_i^\top W_i = \frac{1}{n^k} \sum_{i=1}^{n^k} \text{Tr}(x_i^\top W_i^\top W_i) = \frac{1}{n^k} \sum_{i=1}^{n^k} \text{Tr}(W_i^\top W_i x_i^\top x_i)
  \]
  
  \[
  = \frac{1}{n^k} \text{Tr}(W^\top W X^\top X) = \frac{1}{n^k} \text{Tr}(W^\top W X^\top X)
  \]

- The distance to the hyper-plane (minimized in “synthesis” view):
  
  \[
  \|W X - X\|^2_F = \|X W^\top W - X\|^2_F = \text{Tr}((X W^\top W - X)^\top (X W^\top W - X))
  \]
  
  \[
  = \text{Tr}(W^\top W X^\top X W^\top W - 2 W^\top W X^\top X + X^\top X)
  \]
  
  \[
  = \text{Tr}(W^\top W X^\top X W^\top W) - 2 \text{Tr}(W^\top W X^\top X) + \text{Tr}(X^\top X)
  \]
  
  \[
  = \text{Tr}(W^\top W X^\top X W^\top W) - 2 \text{Tr}(W^\top W X^\top X) + \text{Tr}(X^\top X)
  \]
  
  \[
  = - \text{Tr}(W^\top W X^\top X) + \text{constant}
  \]
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} \left( \left( W_Y^T W_Y \Sigma_{yy}^{1/2} \Sigma_{xy} \Sigma_{xy}^T \Sigma_{yy}^{1/2} \right) \right)
  
  \text{– Subject to } W_X \text{ and } W_Y \text{ having orthogonal rows.}
  \]
• Computationally, equivalent to PCA with a different matrix.
  – Using the “analysis” view that PCA maximizes $\text{Tr}(W^T W X^T X)$. 
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):
  \[ W = UX \]
  \[ \begin{array}{c}
  n \\
  x \\
  d \\
  \end{array} \]
  \[ \begin{array}{c}
  n \\
  x \\
  n \\
  \end{array} \]

• Re-parameterizing in terms of ‘U’ gives a kernelized PCA:
  \[ \text{Tr}(X^T U^T U XX^T X) = \text{Tr}(V^T V XX^T X X^T) \]
  \[ \begin{array}{c}
  K \\
  K \\
  \end{array} \]

• It’s hard to initially center data in ‘Z’ space, but you can form the centered kernel matrix (see here).
Probabilistic PCA

• With zero-mean ("centered") data, in PCA we assume that
  \[ x_i \sim W^T z_i \]

• In probabilistic PCA we assume that
  \[ x_i \sim N(W^T z_i, \sigma^2 I) \]
  \[ z_i \sim N(0, I) \]

• Integrating over ‘Z’ the marginal likelihood given ‘W’ is Gaussian,
  \[ x_i \mid W \sim N(0, W^T W + \sigma^2 I) \]

• Regular PCA is obtained as the limit of \( \sigma^2 \) going to 0.
Generalizations of Probabilistic PCA

• Probabilistic PCA model:

\[ \chi_i \mid W \sim \mathcal{N}(0, W^T W + \sigma^2 I) \]

• 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.

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

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

https://new.edu/resources/big-5-personality-traits
PCA vs. Factor Analysis

• PCA and FA both write the matrix ‘X’ as
  \[ X \approx Z W \]

• 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.
PCA vs. Factor Analysis

- In probabilistic PCA we assume:
  \[ x_i \sim N(\mathbf{W}^\top \mathbf{z}_i, \sigma^2 \mathbf{I}) \]

- In FA we assume for a diagonal matrix \( \mathbf{D} \) that:
  \[ x_i \sim N(\mathbf{W}^\top \mathbf{z}_i, \mathbf{D}) \]

- The posterior in this case is:
  \[ x_i \mid \mathbf{W} \sim N(\mathbf{0}, \mathbf{W}^\top \mathbf{W} + \mathbf{D}) \]

- 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.
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.

http://music.eecs.northwestern.edu/research.php
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 $\alpha$, 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 $Q^TQ = I$.
  - If we rotate $W$ with $Q$, then we have $(QW)^TQW = W^TQ^TQW = W^TW$.

- 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: