Supervised Learning with Tensors

Igor Vorona
Anh-Huy Phan

Skoltech

March 2022

Useful materials

Books

- “Tensor Network for Dimensionality Reduction and Large-Scale Optimizations Part 2 Application and Future Perspectives” (Cichocki et al., 2016), c.2, p.480

Lectures

- “Tensor Methods for Signal Processing and Machine Learning”, Q.Zhao

- “Towards a Regression using Tensors”, Ming Hou

Foundations and Trends® in Machine Learning
Vol. 9, No. 6 (2016) 431–673
© 2017 A. Cichocki *et al.*
DOI: 10.1561/2200000067

now
the essence of knowledge

Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations Part 2 Applications and Future Perspectives

Andrzej Cichocki
Riken, Brain Science Institute, Japan
Skolkovo Institute of Science and Technology (Skoltech), Russia
Systems Research Institute, Polish Academy of Science, Poland
a.cichocki@riken.jp

Anh-Huy Phan
Riken, Brain Science Institute, Japan
phan@brain.riken.jp

Qibin Zhao
Riken Center for Advanced Intelligence Project, Japan
qibin.zhao@riken.jp

Namgil Lee
Kangwon National University, Republic of Korea
namgil.lee@kangwon.ac.kr

Ivan Oseledets
Skolkovo Institute of Science and Technology (Skoltech), Russia
Institute of Numerical Mathematics of Russian Academy of Science
i.oseledets@skolkovotech.ru

Masashi Sugiyama
Riken Center for Advanced Intelligence Project, Japan
University of Tokyo
sugi@ik.u-tokyo.ac.jp

Danilo Mandic
Imperial College, Department of Electrical and Electronic Engineering, UK
d.mandic@imperial.ac.uk

Outline

- Intro
- Regression
- Classification
- Kernel methods

Intro

Supervised Learning with Tensors

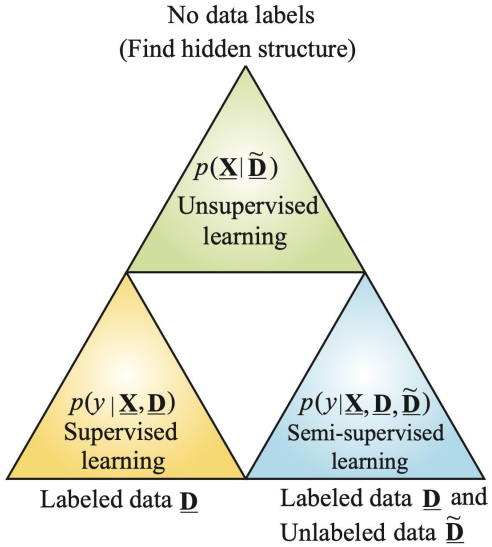
It’s common to use tensor decomposition models in unsupervised setting. For example:

- Dimensionality reduction
- Factor analysis
- Tensor Completion
- ...

But it’s not only one way to work with tensor models

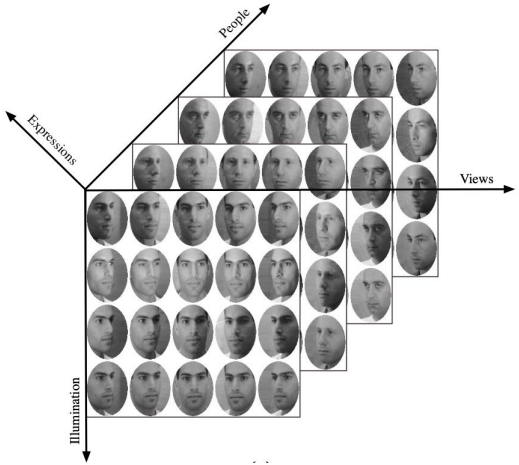
We can work with tensors also in supervised setting.

It’s useful If each element of our dataset represented as tensor.

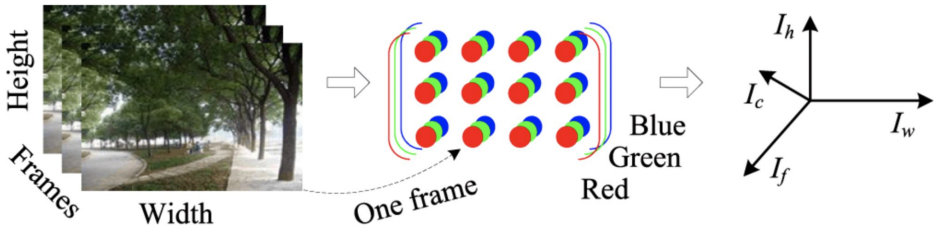


Multidimensional structured data: Computer Vision

- Facial images (expression x people x illumination x views)

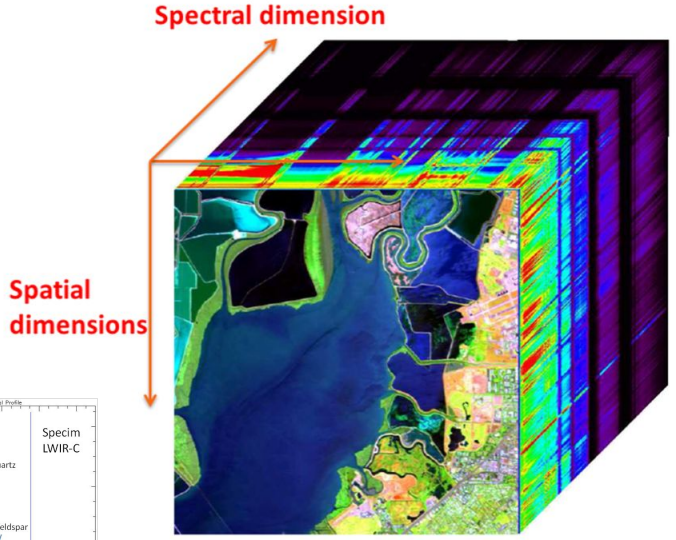


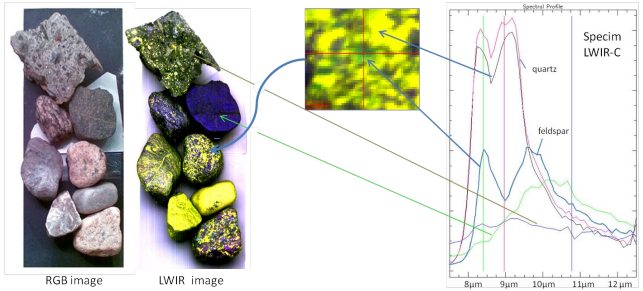
- Video sequences (width x height x channels x frame)



Multidimensional structured data: Hyperspectral imaging

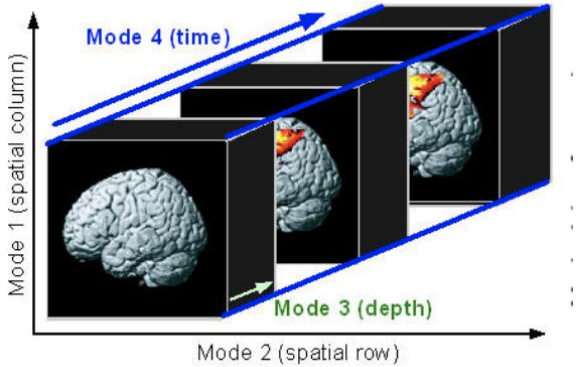
- Hyperspectral imaging is a task of collecting and processing information from across the electromagnetic spectrum
- Each multispectral image can be represented as a three-dimensional data cube, or tensor.
- Examples:
  - Geology
  - Chemistry
  - Agriculture

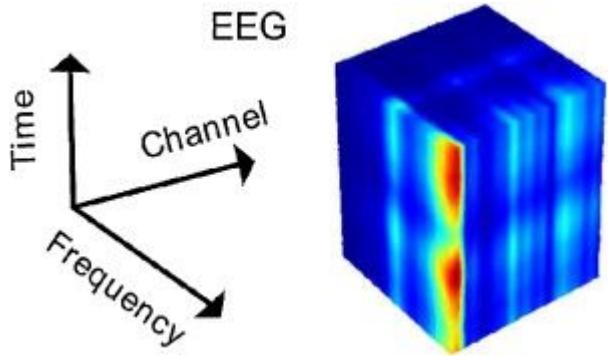




Multidimensional structured data: Biomedicine

- Diagnostics can be based on fMRI (time x 3D volume indexed by cartesian coordinate) and EEG (time x frequency x electrodes)





Practical Tasks

- **Brain Imaging Data Analysis**
  - **Goal** is to find association between brain images (multi-dimensional arrays) and clinical outcomes.
- **Video classification**
  - Predict classes of video frame sequences

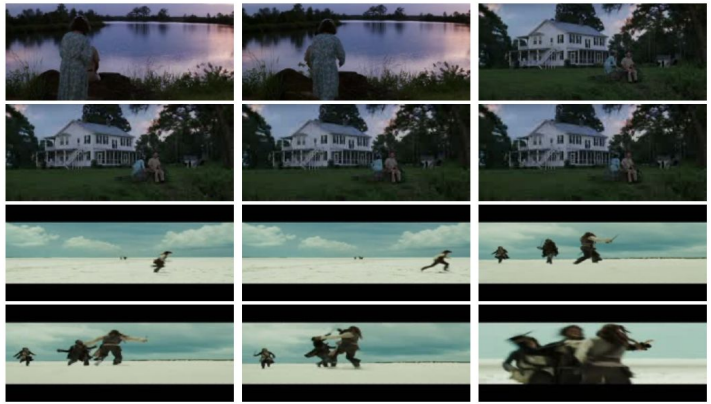


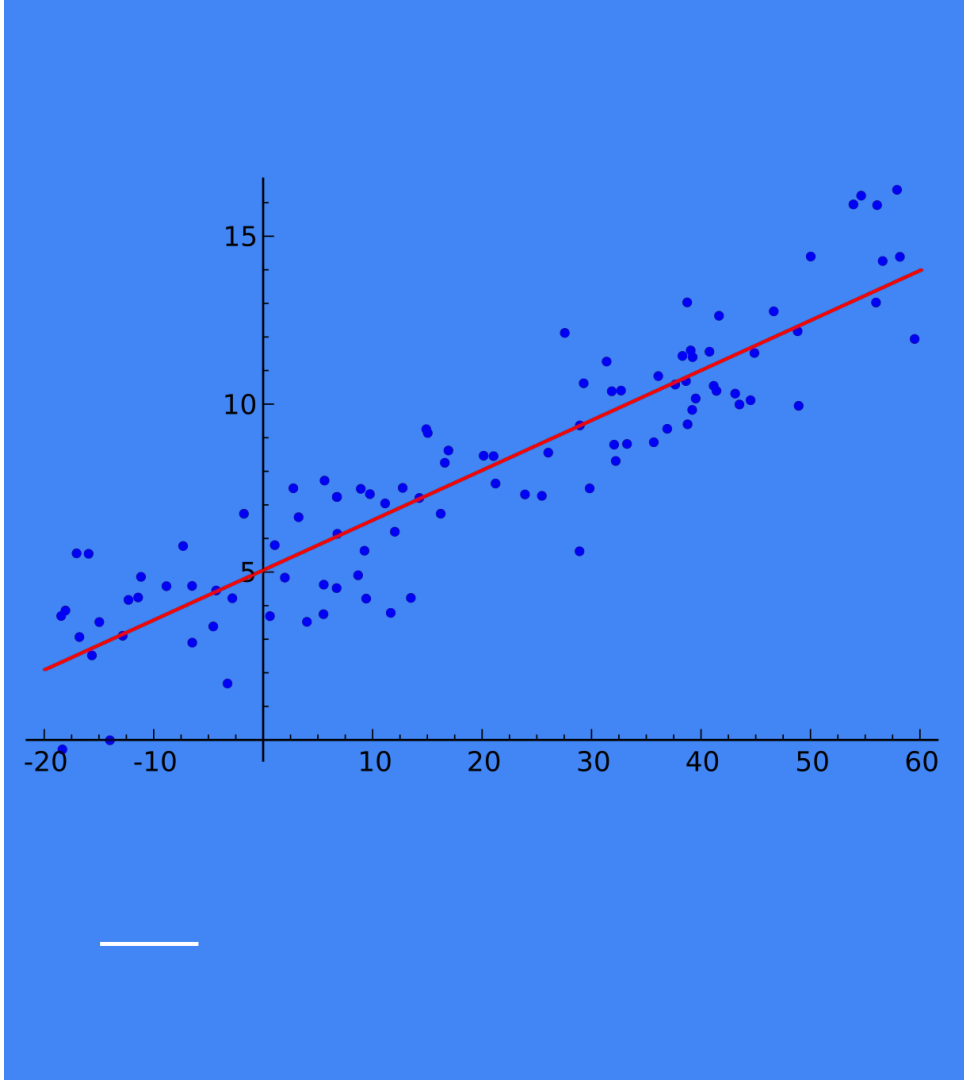
Figure 4: Two samples of frame sequences from the Hollywood2 dataset. The first sequence (row 1 and 2) belongs to the class of sitting down; the second sequence (row 3 and 4) has two labels: running and fighting person.

Advantages of Tensor Representations

We exploit here two main properties of Tensor Representation:

- **Model weights compression**
    - Naive approach works with fMRI images as with 4D array with size 256 x 256 x 256 x 100 (= **167 millions of parameters!).** Tensor based allow to represent dataset in exponentially less number of parameters 256 x R + 256 x R + 256 x R + 100 x R, where R (rank) is some predefined constant.
- **Inductive bias**
    - Tensor representation allow to reveal inherent structure of data and regularize data hypothesis space via specific constraints on the model weights. This trick reduce sample size of the supervised model.

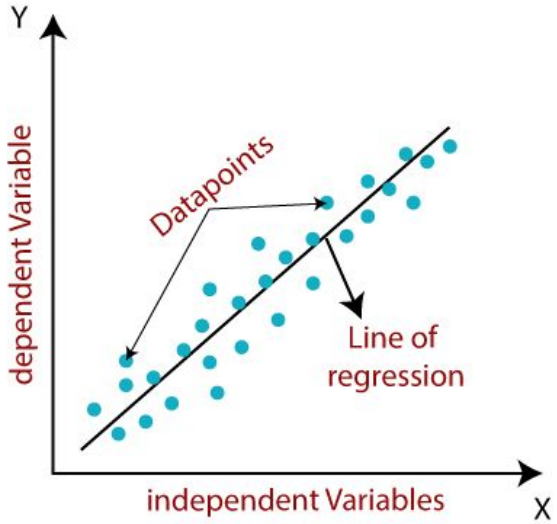
Regression



Regression

- We need to fit curve (in our lecture simple line) to real valued outcomes using several statistical assumption about data. Main assumption is a normal distribution of outcomes given data (Y given X).

$$y = F(\underline{\mathbf{X}}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, 1)$$



Using Maximum Likelihood Estimator of parameters of

our model we have the following problem:

$$J(\underline{\mathbf{X}}, y|\underline{\mathbf{W}}, b) = \sum_{m=1}^{M} (y_m - \langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle - b)^2$$

Where M is size of our dataset.

Basic Tensor Regression Model

$$y = \langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle + b$$

Where

- $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \ldots I_N}$ Is input tensor predictor or tensor regressor
- $\underline{\mathbf{W}} \in \mathbb{R}^{I_1 \times \ldots I_N}$ is the weights of regression coefficient tensor
- $b \in \mathbb{R}$ Is the bias coefficient
- $y \in \mathbb{R}$ Is the regression output or dependent/target variable (response)
- $\langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle = \mathrm{vec}(\underline{\mathbf{X}})^{\mathrm{T}} \mathrm{vec}(\underline{\mathbf{W}})$ Is the inner product of two tensors

CP Regression

We follow basic linear regression model

$$y = \langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle + b$$

But now where

$$\underline{\mathbf{W}} = \sum_{r=1}^{R} u_r^{(1)} \circ u_r^{(2)} \circ \cdots \circ u_r^{(N)} = \underline{\mathbf{I}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}$$

The advantage of CP regression:

- Substantial reduction in dimensionality: 128 x 128 x 128 (MRI image) => the parameters reduce from 2097157 to 1157 via rank-3 decomposition

- Low rank CP model suitable for many low rank signals

Tucker Regression

We follow basic linear regression model

$$y = \langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle + b$$

But now where $\underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}$

Can share properties of CP, but suffer of exponential grow of parameter number of core tensor

The advantage of Tucker regression over CP regression:

- Tucker regression especially useful when data is skewed in dimension (different sizes in modes) by using different ranks of factor matrices
- Tucker regression explicitly model interaction between factor matrices

Fitting of Tensor Regression

Due to multilinearity of tensor decomposition models we can find explicit solution for each factor matrices given other factor matrices.

General idea:

Optimize one factor matrix when other parameters of model are fixed.

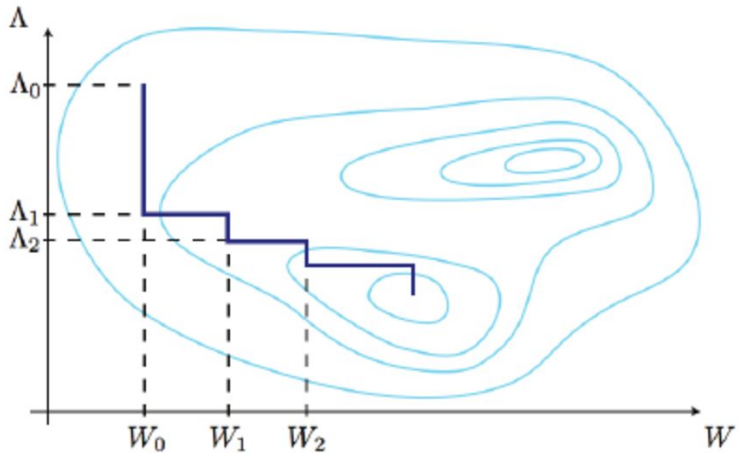
This algorithm called **ALS - alternating least squares.**

Put in X the result of applying parameters (except **w**) to our data

Then solve: $J(\mathbf{Y}, \mathbf{X}; \mathbf{w}) = \|\mathbf{y} - \mathbf{w}^T\mathbf{X}\|_2^2$

$$\frac{dJ}{d\mathbf{w}} = (\mathbf{y} - \mathbf{w}^T\mathbf{X})\mathbf{X}^T = 0$$

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



ALS for CP regression

Our CP model of weights

$$\underline{\mathbf{W}} = \sum_{r=1}^{R} u_r^{(1)} \circ \cdots \circ u^{(N)}$$

Can be express via

$$\mathbf{W}_{(d)} = \mathbf{U}^{(d)} (\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(d+1)} \odot \mathbf{U}^{(d-1)} \odot \cdots \odot \mathbf{U}^{(1)})^T$$

Where $\odot$ means Khatri-Rao product

$$\langle \underline{\mathbf{X}}, \underline{\mathbf{W}} \rangle = \mathrm{tr}(\mathbf{X}_{(d)}^T \mathbf{W}_{(d)}) = \mathrm{tr}(\mathbf{X}_{(d)}^T \mathbf{U}^{(d)} (\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(d+1)} \odot \mathbf{U}^{(d-1)} \odot \cdots \odot \mathbf{U}^{(1)})^T)$$

$$= \mathrm{tr}(\mathbf{U}^{(d)} (\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(d+1)} \odot \mathbf{U}^{(d-1)} \odot \cdots \odot \mathbf{U}^{(1)})^T \mathbf{X}_{(d)}^T) = \left\langle \mathrm{vec}(\mathbf{U}^{(d)}), \mathrm{vec}((\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(d+1)} \odot \mathbf{U}^{(d-1)} \odot \cdots \odot \mathbf{U}^{(1)})^T \mathbf{X}_{(d)}^T) \right\rangle$$

Then for update 1 factor matrix we put $\mathbf{w} = \mathrm{vec}(\mathbf{U}^{(d)})$

And $\mathbf{x} = \mathrm{vec}((\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(d+1)} \odot \mathbf{U}^{(d-1)} \odot \cdots \odot \mathbf{U}^{(1)})^T \mathbf{X}_{(d)}^T)$ column of regressor matrix (1 data element).

Other ways of fitting Tensor Regression

- Any general purpose optimization method like gradient descent
- Tensor Projected Gradient, which combines a gradient step with proximal point projection step with low rank constraints
- ADMM with low-rank constraints
- Riemannian optimization methods

Multilinear Tucker Regression

Assume we have not only tensor-variate regressor, but also dependent variable (response):

$$\{\underline{\mathbf{X}}_m, \underline{\mathbf{Y}}_m\}_{m=1}^{M}$$

which are stacked in concatenated tensors

$$\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times M} \quad \text{and} \quad \underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_N \times M}$$

Multilinear regression model:

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \mathbf{W}_1 \times_2 \mathbf{W}_2 \cdots \times_N \mathbf{W}_N \times_{N+1} \mathbf{D}_M + \underline{\mathbf{E}},$$

Where

$\mathbf{D}_M \in \mathbb{R}^{M \times M}$ (diagonal matrix) $\quad \mathbf{W}_n \in \mathbb{R}^{J_n \times I_n}$ $\quad \underline{\mathbf{E}}$ is zero-mean residual

ALS for Multilinear Tensor Regression

Algorithm 1: Multilinear Tucker Regression

Input: $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times M}$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times I_2 \times \cdots \times J_N \times M}$.

Output: $\{\mathbf{W}_n\}, n = 1, \ldots, N$.

1: Initialize randomly $\mathbf{W}_n$ for $n = 1, \ldots, N$.

2: **while** not converged or iteration limit is not reached **do**

3: **for** $n = 1$ to N **do**

4: $\underline{\mathbf{X}}^{(n)} = \underline{\mathbf{X}} \times_1 \mathbf{W}_1 \cdots \times_{n-1} \mathbf{W}_{n-1} \times_{n+1} \mathbf{W}_{n+1} \cdots \times_N \mathbf{W}_N$

5: Matricize tensors $\underline{\mathbf{X}}^{(n)}$ and $\underline{\mathbf{Y}}$ into their respective unfolded matrices $\mathbf{X}_{(n)}^{(n)}$ and $\mathbf{Y}_{(n)}$

6: Compute $\mathbf{W}_n = \mathbf{Y}_{(n)} (\mathbf{X}_{(n)}^{(n)})^{\mathrm{T}} \left(\mathbf{X}_{(n)}^{(n)} \, (\mathbf{X}_{(n)}^{(n)})^{\mathrm{T}} \right)^{-1}$

7: **end for**

8: **end while**

Regularization of Tensor Regression

Using flexibility of tensors, we can choose different regularization profiles for each slice, fiber and other structures of tensor separately.

- Standard regularization of regression models applied to tensor regression:
  - L2-norm (Tikhonov regularization)

$$\|\underline{\mathbf{W}}\|_F^2 = \langle \underline{\mathbf{W}}, \underline{\mathbf{W}} \rangle$$

  - L1-norm (similar to LASSO)

$$\|\underline{\mathbf{W}}\|_1$$

Tensor rank regularization

- Tensor rank defined naturally in CP format of tensors:

$$\underline{\mathbf{W}} = \sum_{i=1}^{r} \lambda_i u_i^{(1)} \circ u_i^{(2)} \circ \cdots \circ u_i^{(N)}$$

- One of most popular way to minimize instead of the rank it's convex surrogate

$$\|\underline{\mathbf{W}}\|_* = \sum_{n=1}^{N} \|\mathbf{W}_{(n)}\|_*$$

Where $\mathbf{W}_{(n)}$ - n-mode matricization and $\|\underline{\mathbf{W}}\|_* = \sum_{i=1}^{r} \sigma_i$ called *Nuclear* (Trace) norm

Low-rank Regularization via latent trace norm

It has been noticed that previous nuclear norm approach has limitation that it performs poorly for a tensor that is only low-rank in a certain mode.

Some authors proposed an alternative approach, which based on

$$\|\underline{\mathbf{W}}\|_{latent} = \sum_{n=1}^{N} \|\mathbf{W}_{(n)}^{(n)}\|_*$$

$$\text{where } \underline{\mathbf{W}} = \underline{\mathbf{W}}^{(1)} + \underline{\mathbf{W}}^{(2)} + \cdots + \underline{\mathbf{W}}^{(N)}$$

Scaled latent trace norm

If a tensor has a mode with a dimension (size) much smaller than other modes, the latent trace norm may be incorrectly estimated. So, more stable version is *scaled latent trace norm*:

$$\|\underline{\mathbf{W}}\|_{scaled} = \inf_{\underline{\mathbf{W}}} \sum_{n=1}^{N} \frac{1}{\sqrt{I_n}} \|\mathbf{W}_{(n)}^{(n)}\|_*$$

Tensor regression based on the latent trace norm

Note, that latent trace norm induce another tensor representation which based on tensor splitting (additive composition / mixture).

$$\min_{\underline{\mathbf{W}},b} \sum_{m=1}^{M} \left(y_m - \left(\langle \underline{\mathbf{W}}, \underline{\mathbf{X}}_m \rangle + b\right)\right)^2 + \sum_{n=1}^{N} \lambda_n \|\mathbf{W}_{(n)}^{(n)}\|_*$$

$$\text{where } \underline{\mathbf{W}} = \underline{\mathbf{W}}^{(1)} + \underline{\mathbf{W}}^{(2)} + \cdots + \underline{\mathbf{W}}^{(N)}$$

Higher-Order Low-rank Regression (1)

Consider a multivariate regression in which the **response** has a tensor structure.

$$\underline{\mathbf{Y}} = \underline{\mathbf{W}} \,\bar{\times}_1\, \mathbf{x} + \underline{\mathbf{E}}$$

Where:

- $\mathbf{x} \in \mathbb{R}^{I_0}$ is an element of our data (regressor)
- $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a tensor structured response
- $\underline{\mathbf{W}} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$ is a tensor of regression coefficients

Higher-Order Low-rank Regression (2)

Taking fact that all input vector of our dataset can be concatenated into an input matrix

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_M]^T \in \mathbb{R}^{M \times I_0}$$

Optimization problem can be formulated in the following form:

$$\begin{aligned} &\min_{\underline{\mathbf{W}}} \|\underline{\mathbf{W}} \times_1 \mathbf{X} - \underline{\mathbf{Y}}\|_F^2 + \gamma \|\underline{\mathbf{W}}\|_F^2 \\ \text{s.t.} \quad &\underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)}, \\ &\mathbf{U}^{(n)\mathrm{T}} \mathbf{U}^{(n)} = \mathbf{I} \quad \text{for} \quad n = 1, \ldots, N, \end{aligned}$$

where the output tensor **Y** is obtained by stacking the output tensors **Y**m along the first mode

Here we used Tikhonov (L2-norm) regularization of the core tensor + orthogonality constraints of the factor matrices to prevent overfitting and numerical instability.

HOLRR fitting

Our task

$$\min_{\underline{\mathbf{W}}} \|\underline{\mathbf{W}} \times_1 \mathbf{X} - \underline{\mathbf{Y}}\|_F^2 + \gamma \|\underline{\mathbf{W}}\|_F^2$$
$$\text{s.t.} \quad \underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)},$$
$$\mathbf{U}^{(n)\mathrm{T}} \mathbf{U}^{(n)} = \mathbf{I} \quad \text{for} \quad n = 1, \ldots, N,$$

Note, that the regression function can be rewritten in the following map

$$f : \mathbf{x} \mapsto \underline{\mathbf{G}} \times_1 \mathbf{x}^{\mathrm{T}} \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)}$$

And using additionally following substitution $\tilde{\mathbf{X}}^T = (\mathbf{X}|\sqrt{\gamma}\mathbf{I})^T$ and $\tilde{\mathbf{Y}}_{(1)}^T = (\mathbf{Y}_{(1)}|\mathbf{0})^T$

$$\|\underline{\mathbf{W}} \times_1 \mathbf{X} - \underline{\mathbf{Y}}\|_F^2 + \gamma \|\underline{\mathbf{W}}\|_F^2 = \|\underline{\mathbf{W}} \times_1 \tilde{\mathbf{X}} - \underline{\tilde{\mathbf{Y}}}\|_F^2$$

Update rule for core tensor

Let use the following property

$$\text{vec}(\underline{\mathbf{W}}) = (\mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \mathbf{U}^{(0)})\text{vec}(\underline{\mathbf{G}})$$

Then our task is to minimize:

$$\|(\mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \tilde{\mathbf{X}}\mathbf{U}^{(0)})\text{vec}(\underline{\mathbf{G}}) - \text{vec}(\underline{\tilde{\mathbf{Y}}})\|_F^2$$

Let $\mathbf{M} = \mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \tilde{\mathbf{X}}\mathbf{U}^{(0)}$ than solution is given by $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$

Using the mixed-product and inverse properties of the Kronecker product and the column-wise orthogonality of factor matrices

$$\text{vec}(\underline{\mathbf{G}}) = (\mathbf{U}^{(N)} \otimes \cdots \otimes \mathbf{U}^{(1)} \otimes (\mathbf{U}^{(0)T}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\mathbf{U}^{(0)})^{-1}\mathbf{U}^{(0)T}\tilde{\mathbf{X}}^T)\text{vec}(\underline{\tilde{\mathbf{Y}}})$$

$$\underline{\mathbf{G}} = \underline{\mathbf{Y}} \times_1 (\mathbf{U}^{(0)\,\mathrm{T}}(\mathbf{X}^\mathrm{T}\mathbf{X} + \gamma\mathbf{I})\mathbf{U}^{(0)})^{-1}\mathbf{U}^{(0)\,\mathrm{T}}\mathbf{X}^\mathrm{T} \times_2 \mathbf{U}^{(1)\,\mathrm{T}} \cdots \times_{N+1} \mathbf{U}^{(N)\,\mathrm{T}}$$

Update rule for factor matrices (1)

From update rule of core follows:

$$\min_{\substack{\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}, \\ 0 \le i \le p}} \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p - \tilde{\mathcal{Y}}\|_F^2$$

$$\text{s.t. } \mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I} \text{ for } 0 \le i \le p,$$

$$\mathbf{\Pi}_0 = \tilde{\mathbf{X}}\mathbf{U}_0 \left(\mathbf{U}_0 \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \mathbf{U}_0^\top\right)^{-1} \mathbf{U}_0^\top \tilde{\mathbf{X}}^T,$$

$$\mathbf{\Pi}_i = \mathbf{U}_i \mathbf{U}_i^\top \text{ for } 1 \le i \le p \ ,$$

$$\begin{aligned}
\mathcal{L}(\mathcal{W}) &= \|\mathcal{W} \times_1 \tilde{\mathbf{X}} - \tilde{\mathcal{Y}}\|_F^2 \\
&= \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p - \tilde{\mathcal{Y}}\|_F^2 \\
&\le \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p+1} \mathbf{\Pi}_p - \tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_p \mathbf{\Pi}_{p-1}\|_F^2 \\
&+ \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_p \mathbf{\Pi}_{p-1} - \tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 \times_2 \cdots \times_{p-1} \mathbf{\Pi}_{p-2}\|_F^2 \\
&+ \cdots + \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 - \tilde{\mathcal{Y}}\|_F^2 \\
&\le \|\tilde{\mathcal{Y}} \times_{p+1} \mathbf{\Pi}_p - \tilde{\mathcal{Y}}\|_F^2 + \|\tilde{\mathcal{Y}} \times_p \mathbf{\Pi}_{p-1} - \tilde{\mathcal{Y}}\|_F^2 + \cdots + \|\tilde{\mathcal{Y}} \times_1 \mathbf{\Pi}_0 - \tilde{\mathcal{Y}}\|_F^2
\end{aligned}$$

Update rule for factor matrices (2)

Thus we need to solve $\|\underline{\tilde{\mathbf{Y}}} \times_{i+1} \mathbf{\Pi}_i - \underline{\tilde{\mathbf{Y}}}\|_F^2$ for each i

Which is equivalent to minimizing

$$\|\mathbf{\Pi}_i \tilde{\mathbf{Y}}_{(i)}\|_F^2 - 2\left\langle \mathbf{\Pi}_i \tilde{\mathbf{Y}}_{(i)}, \tilde{\mathbf{Y}}_{(i)} \right\rangle = -\|\mathbf{\Pi}_i \tilde{\mathbf{Y}}_{(i)}\|_F^2$$

Due to properties of projection.

And for the factor matrices we use Ri eigenvectors

$$\begin{cases} (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}_{(1)} \tilde{\mathbf{Y}}_{(1)}^\top \tilde{\mathbf{X}} & if\ i = 0 \\ \tilde{\mathbf{Y}}_{(i)} \tilde{\mathbf{Y}}_{(i)}^\top & otherwise \end{cases} \longrightarrow \begin{cases} (\mathbf{X}^\mathrm{T}\mathbf{X} + \gamma \mathbf{I})^{-1} \mathbf{X}^\mathrm{T} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\mathrm{T} \mathbf{X}, & n = 0, \\ \mathbf{Y}_{(n)} \mathbf{Y}_{(n)}^\mathrm{T}, & otherwise. \end{cases}$$

For i > 0 it follows from $\|\mathbf{\Pi}_i \tilde{\mathbf{Y}}_{(i)}\|_F^2 = \mathrm{Tr}(\mathbf{U}^{(i)T} \tilde{\mathbf{Y}}_{(i)} \tilde{\mathbf{Y}}_{(i)}^T \mathbf{U}^{(i)})$

HOLRR ALS

Algorithm 2: Higher-Order Low-Rank Regression (HOLRR)

Input: $\mathbf{X} \in \mathbb{R}^{M \times I_0}$, $\underline{\mathbf{Y}} \in \mathbb{R}^{M \times I_1 \times \cdots \times I_N}$, rank $(R_0, R_1, \ldots, R_N)$ and a regularization parameter γ.

Output: $\underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)}$

1: $\mathbf{U}^{(0)} \leftarrow$ top R_0 eigenvectors of $(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \gamma\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^{\mathrm{T}}\mathbf{X}$

2: **for** $n = 1$ to N **do**

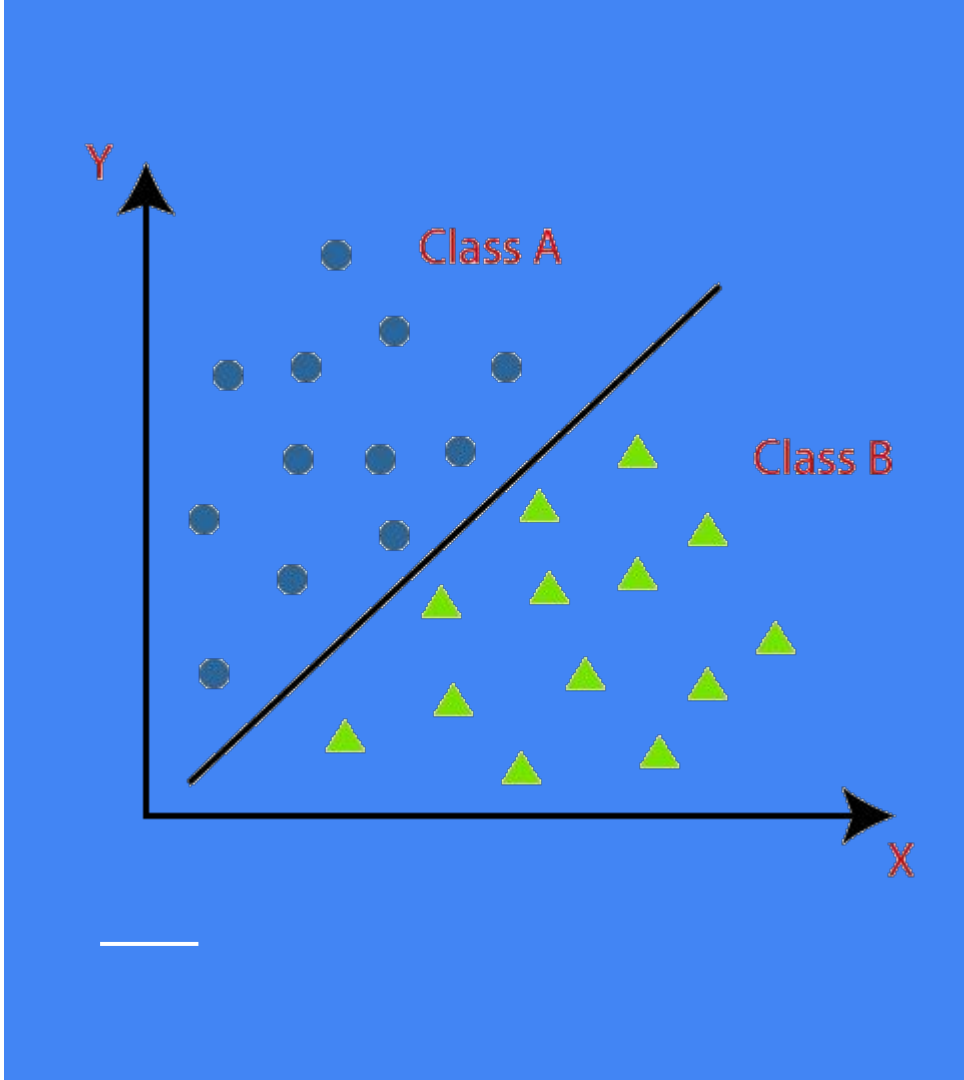
3: $\quad \mathbf{U}^{(n)} \leftarrow$ top R_n eigenvectors of $\mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^{\mathrm{T}}$

4: **end for**

5: $\mathbf{T} = \left(\mathbf{U}^{(0)\,\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \gamma\mathbf{I})\mathbf{U}^{(0)}\right)^{-1}\mathbf{U}^{(0)\,\mathrm{T}}\mathbf{X}^{\mathrm{T}}$

6: $\underline{\mathbf{G}} \leftarrow \underline{\mathbf{Y}} \times_1 \mathbf{T} \times_2 \mathbf{U}^{(1)\,\mathrm{T}} \cdots \times_{N+1} \mathbf{U}^{(N)\,\mathrm{T}}$

Classification



Classification

- Our task to separate two classes

$$\mathcal{X} \in \mathbb{R} \qquad y = (+1, -1)$$

using linear model

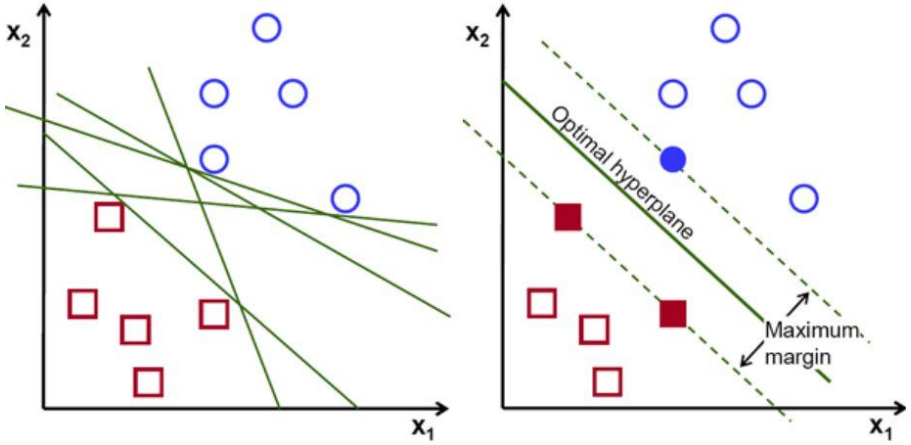
$$y = w^T x + b$$

Optimal separating hyperplane

There are many hyperplanes, which can separate two classes.

Optimal Separating Hyperplane should:

1. Separate the two classes
2. Maximize the distance to the closest point from either class.

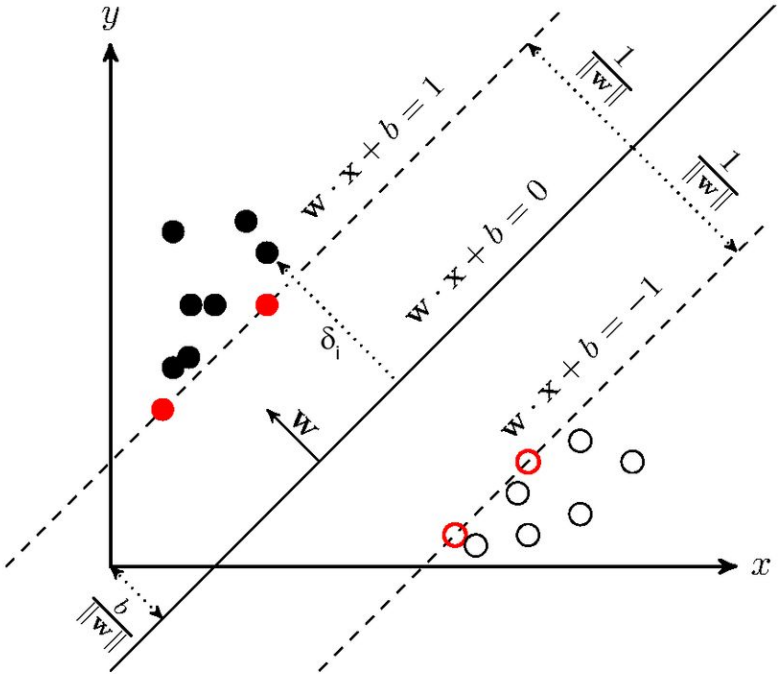


Support Vector Machine

$$\text{minimize}_{w,b} \frac{1}{2} \|w\|_2^2$$

subject to

$$y_i(w^T x_i + b) \geq 1, i = 1, \ldots, m$$



In this case we reweight our problem solutions.

We prefer solution with maximum margin (margin is equal to 1, but margin unit is inverse proportional to ||w||) between two classes.

Soft-SVM

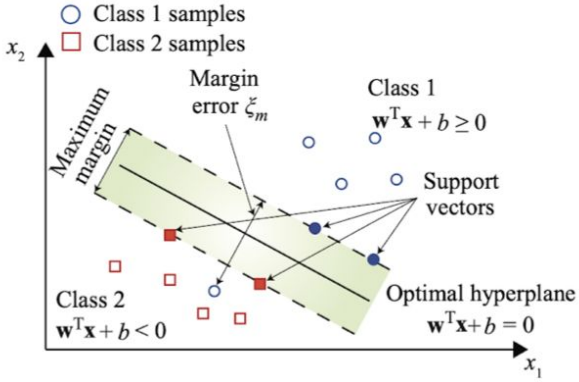
- Our objective function:

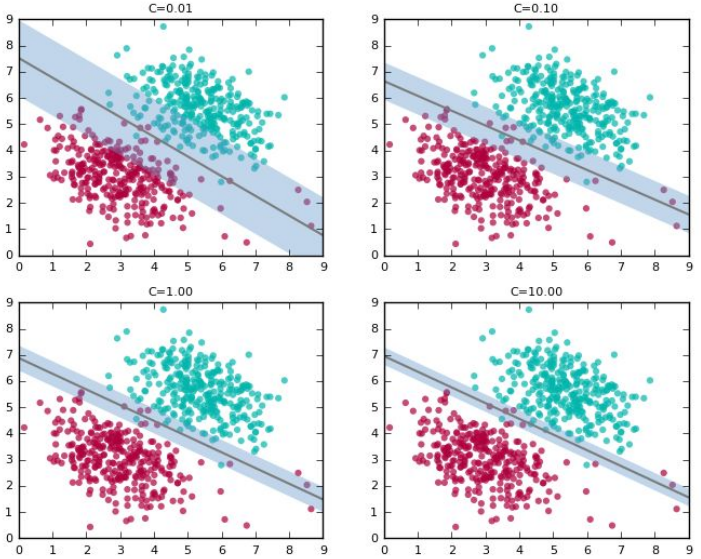
$$\min_{\mathbf{w},b,\boldsymbol{\xi}} J(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{m=1}^{M}\xi_m,$$

$$\text{s.t.} \quad y_m(\mathbf{w}^{\mathrm{T}}\mathbf{x}_m + b) \geqslant 1 - \xi_m, \quad \boldsymbol{\xi} \geqslant 0, \quad m = 1,\ldots,M,$$

When we increase C more ξ to 0's,

SVM becomes more sensitive to data





From SVM to SMM

SMM - **Support Matrix Machines**

This method suitable for matrix data like EEG

For matrices we use frobenius inner product:

$$\langle \mathbf{W}, \mathbf{W} \rangle = \operatorname{tr}\left(\mathbf{W}^T \mathbf{W}\right)$$

Final problem formulation equivalent to original soft-margin SVM + nuclear norm of weights:

$$\begin{aligned} &\arg\min_{\mathbf{W},b,\xi_m} \ \frac{1}{2}\operatorname{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \tau\|\mathbf{W}\|_* + C\sum_{m=1}^{m}\xi_m \\ &\text{s.t.} \quad y_m(\operatorname{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{X}_m) + b) \geqslant 1 - \xi_m. \end{aligned}$$

STM - Support Tensor Machines

Our first generalization SVM to Tensor models

Problem formulation:

$$\min_{\mathbf{w}_n, b, \boldsymbol{\xi}} \; J(\mathbf{w}_n, b, \boldsymbol{\xi}) = \frac{1}{2} \left\| \otimes_{n=1}^{N} \mathbf{w}_n \right\|^2 + C \sum_{m=1}^{M} \xi_m$$

$$\text{s.t.} \quad y_m \left(\underline{\mathbf{X}}_m \bar{\times}_1 \mathbf{w}_1 \cdots \bar{\times}_N \mathbf{w}_N + b \right) \geqslant 1 - \xi_m, \quad \boldsymbol{\xi} \geqslant 0,$$
$$m = 1, \ldots, M.$$

Geometrically STM tries to find N different hyperplanes in N mode spaces

STM Fitting

The STM problem fitted using similar alternating strategy as for other tensor model.

We decompose our original problem by N subproblem and solve it sequentially one by one.

The n-th QP sub-problem formulated, as follows

$$\min_{\mathbf{w}_n, b, \boldsymbol{\xi}} \ \frac{1}{2} \left\| \mathbf{w}_n \right\|^2 \prod_{1 \leqslant i \leqslant N}^{i \neq n} \left(\left\| \mathbf{w}_i \right\|^2 \right) + C \sum_{m=1}^{M} \xi_m$$

$$\text{s.t.} \quad y_m \left(\mathbf{w}_n^{\mathrm{T}} \left(\underline{\mathbf{X}}_m \bar{\times}_{i \neq n} \mathbf{w}_i \right) + b \right) \geqslant 1 - \xi_m, \quad \boldsymbol{\xi} \geqslant 0,$$

$$m = 1, \ldots, M.$$

HRSTM - Higher Rank STM

Higher Rank STM aim to estimate weights of SVM in CPD format. Construct hyperplane in the tensor space.

Problem formulation:

$$\min_{\underline{\mathbf{W}},b,\boldsymbol{\xi}} \ \frac{1}{2}\langle \underline{\mathbf{W}}, \underline{\mathbf{W}} \rangle + C \sum_{m=1}^{M} \xi_m,$$

$$\text{s.t.} \quad y_m \left(\langle \underline{\mathbf{W}}, \underline{\mathbf{X}}_m \rangle + b \right) \geqslant 1 - \xi_m, \quad \xi_m \geqslant 0, \quad m = 1, \ldots, M,$$

where

$$\underline{\mathbf{W}} = \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \circ \mathbf{u}_r^{(2)} \circ \cdots \circ \mathbf{u}_r^{(N)}$$

And

$$\langle \underline{\mathbf{W}}, \underline{\mathbf{W}} \rangle = \operatorname{tr}[\mathbf{W}_{(n)} \mathbf{W}_{(n)}^{\mathrm{T}}] = \operatorname{vec}(\mathbf{W}_{(n)})^{\mathrm{T}} \operatorname{vec}(\mathbf{W}_{(n)})$$

HRSTM Fitting via SMM

General idea comes again from ALS: At each iteration we solve only for the parameters that are associated with the n-th mode of the parameter tensor, while keeping all the other parameters fixed.

We use following property

$$\mathbf{W}_{(n)} = \mathbf{U}^{(n)}(\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \cdots \odot \mathbf{U}^{(1)})^{\mathrm{T}} = \mathbf{U}^{(n)}(\mathbf{U}^{(-n)})^{\mathrm{T}}$$

Then formulate problem in the SMM way and then each of N subproblems solved via QP:

$$\begin{aligned} &\min_{\mathbf{U}^{(n)},b,\boldsymbol{\xi}} \ \frac{1}{2}\mathrm{tr}(\mathbf{U}^{(n)}(\mathbf{U}^{(-n)})^{\mathrm{T}}(\mathbf{U}^{(-n)})(\mathbf{U}^{(n)})^{\mathrm{T}}) + C\sum_{m=1}^{M}\xi_m, \\ &\text{s.t.} \quad y_m\left(\mathrm{tr}\left(\mathbf{U}^{(n)}(\mathbf{U}^{(-n)})^{\mathrm{T}}\mathbf{X}_{m(n)}^{\mathrm{T}}\right) + b\right) \geqslant 1 - \xi_m, \quad \xi_m \geqslant 0, \\ &\qquad m = 1, \ldots, M, \quad n = 1, \ldots, N. \end{aligned}$$

HRSTM Fitting via SVM

Additionally we can use SVM (instead of SMM) formulation of Higher Rank STM problem.

Then we use following trick.

Let

$$\mathbf{A} = \mathbf{U}^{(-n)^{\mathrm{T}}}\mathbf{U}^{(-n)}$$

$$\widetilde{\mathbf{U}}^{(n)} = \mathbf{U}^{(n)}\mathbf{A}^{\frac{1}{2}}$$

Then

$$\begin{aligned}\mathrm{tr}[\mathbf{U}^{(n)}(\mathbf{U}^{(-n)})^{\mathrm{T}}(\mathbf{U}^{(-n)})(\mathbf{U}^{(n)})^{\mathrm{T}}] &= \mathrm{tr}[\widetilde{\mathbf{U}}^{(n)}(\widetilde{\mathbf{U}}^{(n)})^{\mathrm{T}}] \\ &= \mathrm{vec}(\widetilde{\mathbf{U}}^{(n)})^{\mathrm{T}}\mathrm{vec}(\widetilde{\mathbf{U}}^{(n)}).\end{aligned}$$

HRSTM Fitting via SVM

Finally we solve following sequence of SVM tasks for each $\tilde{\mathbf{U}}^{(n)}$

$$\begin{aligned} &\min_{\mathbf{U}^{(n)},b,\boldsymbol{\xi}} \frac{1}{2}\mathrm{vec}(\tilde{\mathbf{U}}^{(n)})^{\mathrm{T}}\mathrm{vec}(\tilde{\mathbf{U}}^{(n)}) + C\sum_{m=1}^{M}\xi_m, \\ \text{s.t.} \quad & y_m\left(\mathrm{vec}(\tilde{\mathbf{U}}^{(n)})^{\mathrm{T}}\mathrm{vec}(\tilde{\mathbf{X}}_{m(n)}) + b\right) \geqslant 1-\xi_m, \quad \xi_m \geqslant 0, \\ & m = 1,\ldots,M, \quad n = 1,\ldots,N. \end{aligned}$$

And after each $\tilde{\mathbf{U}}^{(n)}$ update we go to original parameters via inverse transform

$$\mathbf{U}^{(n)} = \tilde{\mathbf{U}}^{(n)}\mathbf{A}^{-\frac{1}{2}}$$

Fisher Discriminant Analysis

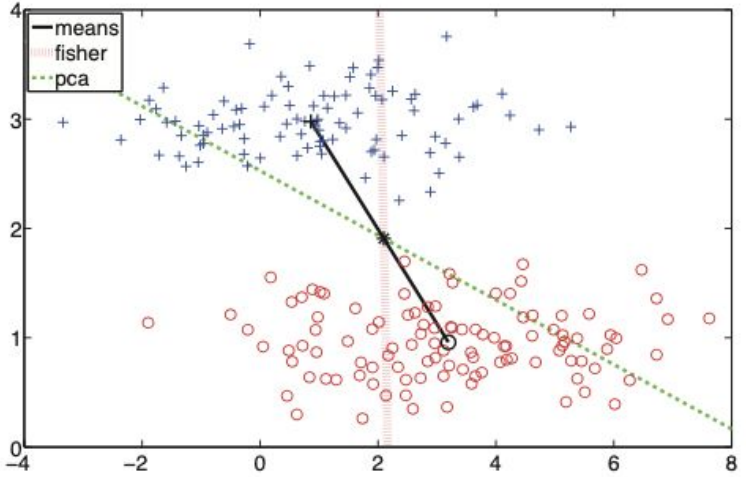
$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{i:y_i=1} \mathbf{x}_i, \; \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{i:y_i=2} \mathbf{x}_i$$

Let

$$m_k = \mathbf{w}^T \boldsymbol{\mu}_k \qquad s_k^2 = \sum_{i:y_i=k} (z_i - m_k)^2$$

$$z_i = \mathbf{w}^T \mathbf{x}_i$$

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \longrightarrow J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$



where $\mathbf{S}_B$ is the between-class scatter matrix given by

$$\mathbf{S}_B = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T$$

and $\mathbf{S}_W$ is the within-class scatter matrix, given by

$$\mathbf{S}_W = \sum_{i:y_i=1} (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T + \sum_{i:y_i=2} (\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^T$$

Tensor Fisher Discriminant Analysis

The decision function is a multilinear function

$$y(\underline{\mathbf{X}}) = \text{sign}(\underline{\mathbf{X}} \bar{\times}_1 \mathbf{w}_1 \cdots \bar{\times}_N \mathbf{w}_N + b)$$

Let

$$\underline{\mathbf{L}}_+ = (1/M_+) \sum_{m=1}^{M} (I(y_m = +1) \underline{\mathbf{X}}_m)$$

$$\underline{\mathbf{L}}_- = (1/M_-) \sum_{m=1}^{M} (I(y_m = -1) \underline{\mathbf{X}}_m)$$

$$\underline{\mathbf{L}} = (1/M) \sum_{m=1}^{M} \underline{\mathbf{X}}_m$$

In TFDA our problem become:

$$\max_{\mathbf{w}_n|_{n=1}^{N}} J(\mathbf{w}_n) = \frac{\|(\underline{\mathbf{L}}_+ - \underline{\mathbf{L}}_-) \bar{\times}_1 \mathbf{w}_1 \cdots \bar{\times} \mathbf{w}_N\|^2}{\sum_{m=1}^{M} \|(\underline{\mathbf{X}}_m - \underline{\mathbf{L}}) \bar{\times}_1 \mathbf{w}_1 \cdots \bar{\times} \mathbf{w}_N\|^2}$$

Multiway Fisher Discriminant Analysis

The decision function is again linear like in the FDA, but weights will be decomposed

$$\mathbf{w} = \mathbf{w}_N \otimes \cdots \otimes \mathbf{w}_1$$

Let $\mathbf{X}_{(1)} \in \mathbb{R}^{M \times I_1 I_2 \cdots I_N}$ be an unfolded matrix of observed samples

$\mathbf{Y} \in \mathbb{R}^{M \times C}$ Is the matrix of dummy variables indicating the group memberships

In MFDA our problem become:

$$\arg\max_{\mathbf{w}} \frac{\mathbf{w}^{\mathrm{T}} \mathbf{S}_B \mathbf{w}}{\mathbf{w}^{\mathrm{T}} \mathbf{S}_T \mathbf{w} + \lambda \mathbf{w}^{\mathrm{T}} \mathbf{R} \mathbf{w}}$$

$$\mathbf{S}_B = \mathbf{X}_{(1)}^{\mathrm{T}} \mathbf{Y} (\mathbf{Y}^{\mathrm{T}} \mathbf{Y})^{-1} \mathbf{Y}^{\mathrm{T}} \mathbf{X}_{(1)}$$

$$\mathbf{S}_T = \mathbf{X}_{(1)}^{\mathrm{T}} \mathbf{X}_{(1)}$$

Multilinear Discriminant Analysis

Suppose that we have M tensor samples

$$\underline{\mathbf{X}}_m \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, m = 1, \ldots, M$$

Belonging to one of the C classes

$$\underline{\mathbf{Z}}_m = \underline{\mathbf{X}}_m \times_1 \mathbf{W}_1^{\mathrm{T}} \cdots \times_N \mathbf{W}_N^{\mathrm{T}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$$

Let

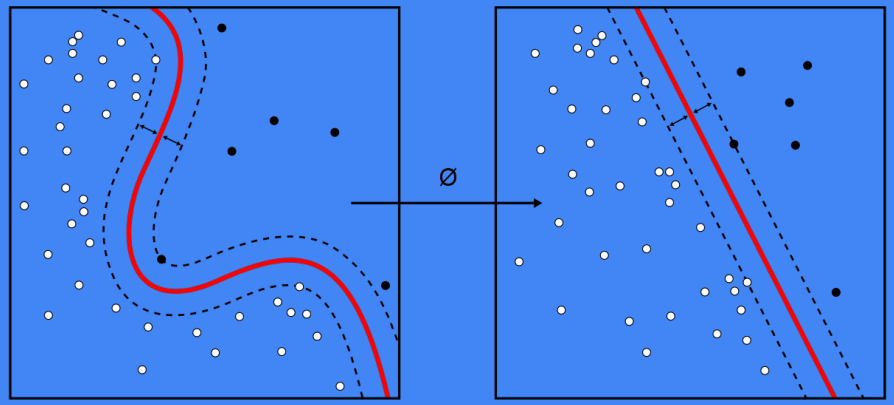
$$\underline{\mathbf{L}}_c = \frac{1}{M_c} \sum_{m=1}^{M_c} \underline{\mathbf{X}}_m I(y_m = c) \qquad \widetilde{\underline{\mathbf{L}}}_c = \frac{1}{M_c} \sum_{m=1}^{M_c} \underline{\mathbf{Z}}_m I(y_m = c) = \underline{\mathbf{L}}_c \times_1 \mathbf{W}_1^{\mathrm{T}} \cdots \times_N \mathbf{W}_N^{\mathrm{T}}$$

$$\underline{\mathbf{L}} = \frac{1}{M} \sum_{m=1}^{M} \underline{\mathbf{X}}_m$$

In MDA our problem become:

$$J(\mathbf{W}_n) = \frac{\sum_{c=1}^{C} M_c \| (\underline{\mathbf{L}}_c - \underline{\mathbf{L}}) \prod_{n=1}^{N} \times_n \mathbf{W}_n^{\mathrm{T}} \|_F^2}{\sum_{c=1}^{C} \sum_{m=1}^{M_c} \| (\underline{\mathbf{X}}_m^c - L_c) \prod_{n=1}^{N} \times_n \mathbf{W}_n^{\mathrm{T}} \|_F^2} = \frac{\mathrm{Tr}(\mathbf{W}_n^{\mathrm{T}} \mathbf{B}_{-n} \mathbf{W}_n)}{\mathrm{Tr}(\mathbf{W}_n^{\mathrm{T}} \mathbf{S}_{-n} \mathbf{W}_n)},$$

Kernel methods



Kernel trick

- Kernel trick allow to solve nonlinear problem by linear model using special function called kernels

- We can rewrite our linear model in dual form as combination of inner products between each pair of elements in dataset

- Kernel is nonlinear analog of inner products between vectors

$$k : [a, b] \times [a, b] \rightarrow \mathbb{R}$$

Main property of kernel function is **positive definiteness** of matrix of pairwise inner products. Only in this positive definite case kernel can induce unique Hilbert Space.

Kernel trick

General form of kernel:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Where ϕ is nonlinear projection

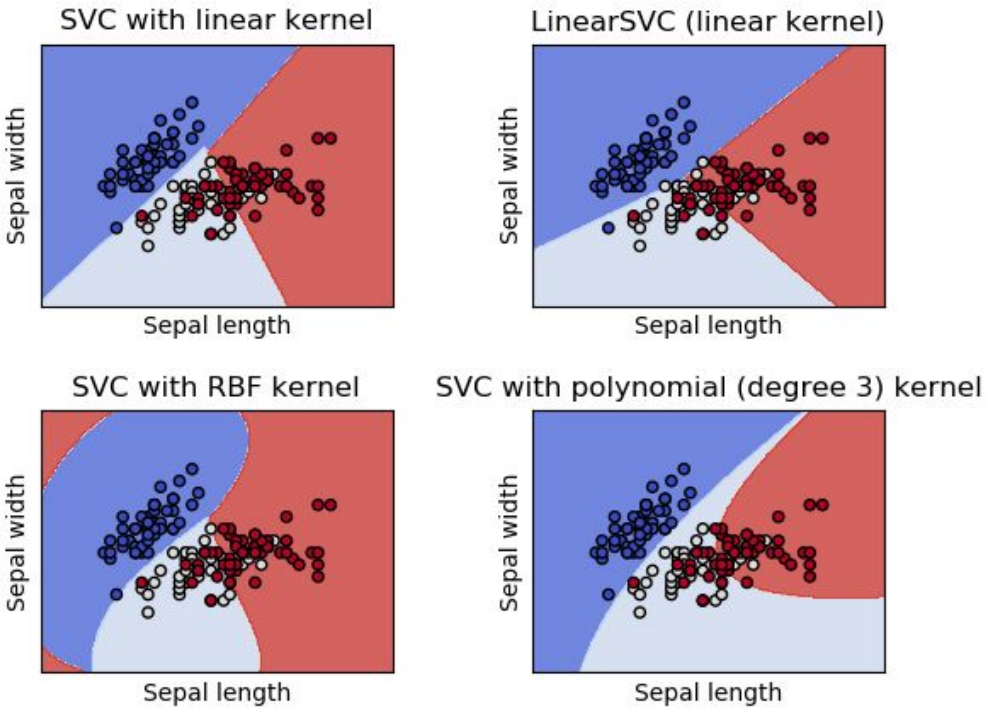
Examples:

- Polynomial

$$k(x_i, x_j) = (a x_i^T x_j + b)^d$$

- Radial basis function

$$k(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|_2^2}{\sigma^2}\right)$$



Kernel function in Tensor Learning

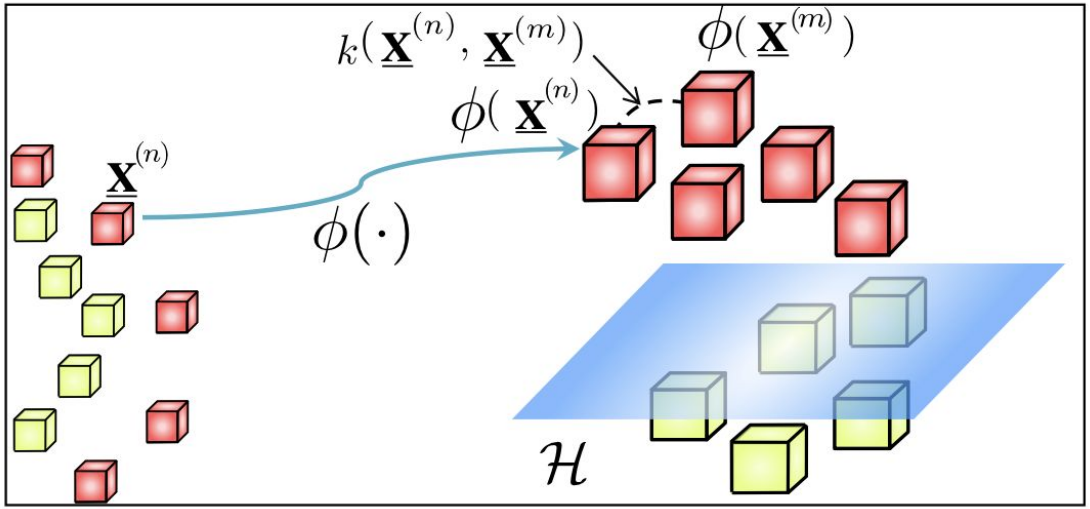


Figure 2.5: The concept of kernel learning for tensors. Tensor observations are mapped into the reproducing kernel Hilbert space $\mathcal{H}$ by a nonlinear mapping function $\phi(\cdot)$. The kernel function serves as a similarity measure between two tensors.

Kernel function in Tensor Learning

kernel functions $k : \underline{\mathbf{X}} \times \underline{\mathbf{X}} \to \mathbb{R}$, given by

Linear kernel: $k(\underline{\mathbf{X}}, \underline{\mathbf{X}}') = \langle \mathrm{vec}(\underline{\mathbf{X}}), \mathrm{vec}(\underline{\mathbf{X}}') \rangle,$

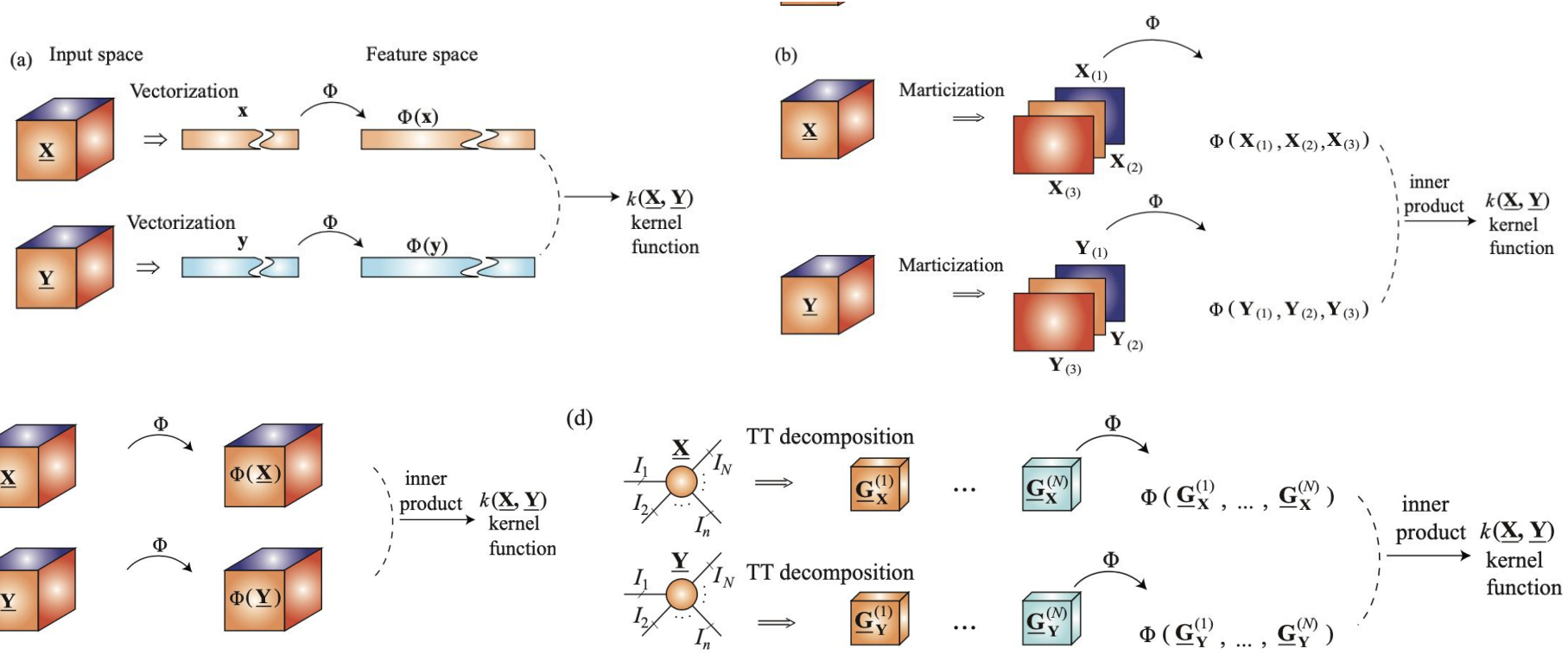
Gaussian-RBF kernel: $k(\underline{\mathbf{X}}, \underline{\mathbf{X}}') = \exp\left(-\frac{1}{2\beta^2} \|\underline{\mathbf{X}} - \underline{\mathbf{X}}'\|_F^2 \right)$

Probabilistic product kernel:

$$k(\underline{\mathbf{X}}, \underline{\mathbf{X}}') = \alpha^2 \prod_{n=1}^{N} \exp\left(-\frac{1}{2\beta_n^2} S_n(\underline{\mathbf{X}} \| \underline{\mathbf{X}}') \right),$$

$$S_n(\underline{\mathbf{X}} \| \underline{\mathbf{X}}') = D\left(p(\mathbf{x}|\mathbf{\Omega}_n^{\underline{\mathbf{X}}}) \| q(\mathbf{x}|\mathbf{\Omega}_n^{\underline{\mathbf{X}}'}) \right)$$

Constructing tensor kernel functions



Tensor manifold kernels

Grassmann manifold of linear space is a manifold of p-dimensional linear subspaces.

In this case by using Grassmann manifold constraints we constrain rank of each n-mode matricization of a tensor.

Chordal distance on Grassmann manifold is

$$k(\underline{\mathbf{X}}, \underline{\mathbf{X}'}) = \prod_{n=1}^{N} \exp\left(-\frac{1}{2\beta^2}\left\|\mathbf{V}_{\mathbf{X}}^{(n)}\mathbf{V}_{\mathbf{X}}^{(n)\,\mathrm{T}} - \mathbf{V}_{\mathbf{X}'}^{(n)}\mathbf{V}_{\mathbf{X}'}^{(n)\,\mathrm{T}}\right\|_F^2\right)$$

where

$$\mathbf{X}_{(n)} = \mathbf{U}_{\mathbf{X}}^{(n)}\mathbf{\Sigma}_{\mathbf{X}}^{(n)}\mathbf{V}_{\mathbf{X}}^{(n)\,\mathrm{T}}$$

How to apply kernels?

- Kernels usually applied for data tensors $\underline{\mathbf{X}}$ rather than weight tensors $\underline{\mathbf{W}}$
- For SVM problems we use svm dual reformulation
- For Frobenius norm minimization we start from existence of nonlinear mapping from our initial vector space to RKHS (space induced by kernels).

  Assume we know how to do it and then put instead of $\mathbf{X}_{(n)} \rightarrow \Phi(\mathbf{X}_{(n)})$

  And then when we see $\mathbf{X}_{(n)}^T \mathbf{X}_{(n)}$ in our equation we put instead kernel matrix

$$K(\mathbf{X}_{(n)}, \mathbf{X}_{(n)}) = \Phi(\mathbf{X}_{(n)})^T \Phi(\mathbf{X}_{(n)})$$

Kernel HOLRR

Let $\phi : \mathbb{R}^{I_0} \to \mathbb{R}^{L}$ be a feature map

Let $\mathbf{\Phi} \in \mathbb{R}^{M \times L}$ be a matrix with row vectors $\phi(\mathbf{x}_m^{\mathrm{T}})$ for $m \in \{1, \ldots, M\}$

$$\min_{\underline{\mathbf{W}}} \|\underline{\mathbf{W}} \times_1 \mathbf{\Phi} - \underline{\mathbf{Y}}\|_F^2 + \gamma \|\underline{\mathbf{W}}\|_F^2$$

$$\text{s.t.} \quad \underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)},$$

$$\text{and} \quad \mathbf{U}^{(n)\mathrm{T}} \mathbf{U}^{(n)} = \mathbf{I} \quad \text{for} \quad n = 0, \ldots, N.$$

Ordinary HOLRR

Algorithm 2: Higher-Order Low-Rank Regression (HOLRR)

Input: $\mathbf{X} \in \mathbb{R}^{M \times I_0}$, $\underline{\mathbf{Y}} \in \mathbb{R}^{M \times I_1 \times \cdots \times I_N}$, rank $(R_0, R_1, \ldots, R_N)$ and a regularization parameter γ.

Output: $\underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)}$

1: $\mathbf{U}^{(0)} \leftarrow$ top R_0 eigenvectors of $(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \gamma\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^{\mathrm{T}}\mathbf{X}$

2: **for** $n = 1$ to N **do**

3: $\quad \mathbf{U}^{(n)} \leftarrow$ top R_n eigenvectors of $\mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^{\mathrm{T}}$

4: **end for**

5: $\mathbf{T} = \left(\mathbf{U}^{(0)\,\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \gamma\mathbf{I})\mathbf{U}^{(0)}\right)^{-1}\mathbf{U}^{(0)\,\mathrm{T}}\mathbf{X}^{\mathrm{T}}$

6: $\underline{\mathbf{G}} \leftarrow \underline{\mathbf{Y}} \times_1 \mathbf{T} \times_2 \mathbf{U}^{(1)\,\mathrm{T}} \cdots \times_{N+1} \mathbf{U}^{(N)\,\mathrm{T}}$

Updates of Kernelized HOLRR

We need to compute top R eigenvectors from

$$(\mathbf{\Phi}^\top \mathbf{\Phi} + \gamma \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi}$$

Let $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ is a Gram matrix

Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ is an eigenvector of $(\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K}$

Then $\mathbf{v} = \mathbf{\Phi}^\top \boldsymbol{\alpha} \in \mathbb{R}^L$ is an appropriate eigenvector:

$$\begin{aligned}
\lambda \mathbf{v} &= \mathbf{\Phi}^\top (\lambda \boldsymbol{\alpha}) = \mathbf{\Phi}^\top \left((\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{K} \right) \boldsymbol{\alpha} \\
&= \mathbf{\Phi}^\top (\mathbf{\Phi}\mathbf{\Phi}^\top + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \boldsymbol{\alpha} \\
&= \left((\mathbf{\Phi}^\top \mathbf{\Phi} + \gamma \mathbf{I})^{-1} \mathbf{\Phi}^\top \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^\top \mathbf{\Phi} \right) \mathbf{v} \ .
\end{aligned}$$

Updates of Kernelized HOLRR

Let $\mathbf{U}^{(0)}$ be the top R eigenvectors of the matrix $(\mathbf{K}+\gamma\mathbf{I})^{-1}\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^{\top}\mathbf{K}$

Then $\mathbf{U}^{(0)}_{ordinary} = \mathbf{\Phi}^{T}\mathbf{U}^{(0)}_{kernalized}$

To resolve this problem we define another regression function:

$$f : \mathbf{x} \mapsto \underline{\mathbf{W}} \bar{\times}_1 \mathbf{k}_x = \sum_{m=1}^{M} k(\mathbf{x}, \mathbf{x}_m)\underline{\mathbf{W}}_m$$

and

$$\mathbf{T} = (\mathbf{U}^{(0)T}\mathbf{\Phi}(\mathbf{\Phi}^{T}\mathbf{\Phi}+\gamma\mathbf{I})\mathbf{\Phi}^{T}\mathbf{U}^{(0)})^{-1}\mathbf{U}^{(0)T}\mathbf{\Phi}\mathbf{\Phi}^{T} = (\mathbf{U}^{(0)T}\mathbf{K}(\mathbf{K}+\gamma\mathbf{I})\mathbf{U}^{(0)})^{-1}\mathbf{U}^{(0)T}\mathbf{K}$$

Kernelized HOLRR

Algorithm 3: Kernelized Higher-Order Low-Rank Regression (KHOLRR)

Input: Gram matrix $\mathbf{K} \in \mathbb{R}^{M \times M}$, $\underline{\mathbf{Y}} \in \mathbb{R}^{M \times I_1 \times \cdots \times I_N}$, rank $(R_0, R_1, \ldots, R_N)$ and a regularization parameter γ.

Output: $\underline{\mathbf{W}} = \underline{\mathbf{G}} \times_1 \mathbf{U}^{(0)} \times_2 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N)}$

1: $\mathbf{U}^{(0)} \leftarrow$ top R_0 eigenvectors of $(\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{Y}_{(1)} \mathbf{Y}_{(1)}^{\mathrm{T}} \mathbf{K}$

2: **for** $n = 1$ to N **do**

3: $\quad \mathbf{U}^{(n)} \leftarrow$ top R_n eigenvectors of $\mathbf{Y}_{(n)} \mathbf{Y}_{(n)}^{\mathrm{T}}$

4: **end for**

5: $\mathbf{T} = \left(\mathbf{U}^{(0)\,\mathrm{T}} \mathbf{K} (\mathbf{K} + \gamma \mathbf{I}) \mathbf{U}^{(0)} \right)^{-1} \mathbf{U}^{(0)\,\mathrm{T}} \mathbf{K}$

6: $\underline{\mathbf{G}} \leftarrow \underline{\mathbf{Y}} \times_1 \mathbf{T} \times_2 \mathbf{U}^{(1)\,\mathrm{T}} \cdots \times_{N+1} \mathbf{U}^{(N)\,\mathrm{T}}$

Thank you for your attention! :)