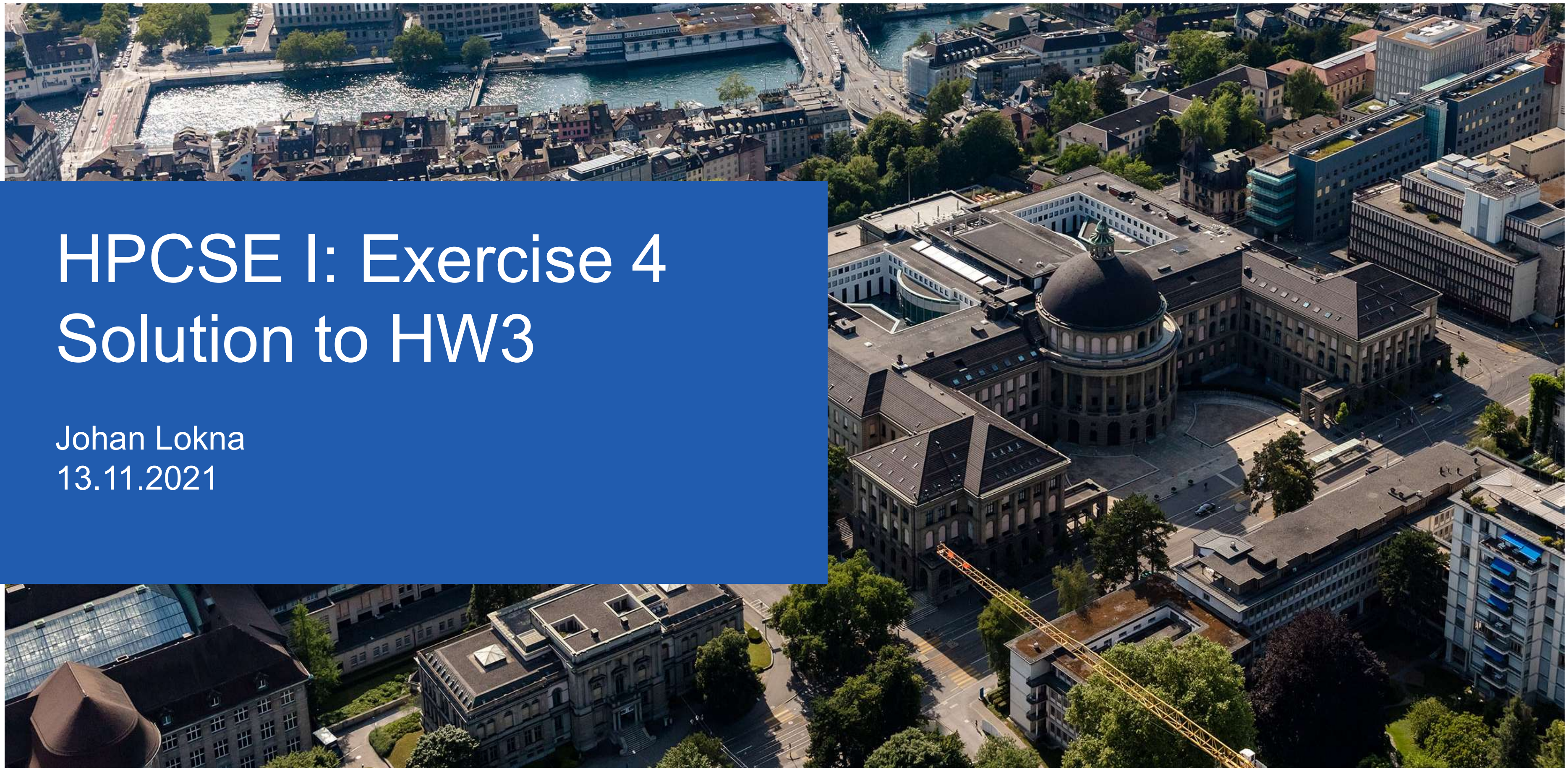
Computational Science
and Engineering Lab



HPCSE I: Exercise 4
Solution to HW3

Johan Lokna

13.11.2021

Agenda

1. **Solution to HW 3**
   a) Question 1 – Perceptron
   b) Question 2 – Linear Auto-Encoders
2. **Introduction to HW 4**

Question 1a – Formulation

In this question we want to investigate methods for computing the Eigen values and vectors of a dataset's co-variance matrix:

$$\mathbf{C} := \frac{1}{N-1}\mathbf{X}^T\mathbf{X}, \text{ with } \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \dots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times n} \tag{1}$$

All methods are based on the Perceptron, but use different rules for updating the weights. Importantly, the updates strongly depend on the output from the forward-pass of the Perceptron, as defined in equation (2):

$$\mathbf{O} = \mathbf{X}\mathbf{W} \text{ , with } \mathbf{W} \in \mathbb{R}^{n\times k} \tag{2}$$

[5 points] Implement the forward pass of the Perceptron in the file *perceptron.cpp*. Use a single call to BLAS in your implementation.

Question 1a – Solution

```
void Perceptron::forward(const scalar* data, scalar* out, const int batch_size) {

    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
                batch_size, out_dims_, in_dims_,
                1.0, data, in_dims_, weights_, out_dims_,
                0.0, out, out_dims_
    );

}
```

Question 1b – Formulation

b) **[20 points]** In order to find Eigen values and vectors, we need to consider the evolution of our weights. We define $\mathbf{O}^t$ to be the output matrix from step t.

$$\mathbf{O}^t = \left[\mathbf{o}_1^t \ldots \mathbf{o}_n^t\right], \text{ such that } \mathbf{o}_i^t = \mathbf{X}\mathbf{w}_i^t \tag{3}$$

As indicated, we see that our weights evolve over time and $\mathbf{w}_i^t$ is the i-th weight vector at step t. We then need to define the gradient update rules, which is done in the following manner:

3. Sangers's Rule:

$$\begin{aligned}\nabla \mathbf{w}_i^t &\propto \mathbf{X}^T \mathbf{o}_i^t - \sum_{j=1}^{i} \left(\left(\mathbf{o}_i^t\right)^T \mathbf{o}_j^t\right) \mathbf{w}_j^t \\ &= \mathbf{X}^T \mathbf{o}_i^t - \left[\mathbf{w}_1 \ldots \mathbf{w}_i\right] \begin{bmatrix} \mathbf{o}_1^T \\ \ldots \\ \mathbf{o}_i^T \end{bmatrix} \mathbf{o}_i \end{aligned} \tag{6}$$

Your task is to implement Sanger's update rule in *sanger.cpp* as given by equation (6). Also in this task, each TO DO corresponds to a single function call; either to BLAS or another class method.

Question 1b – Solution

```
void SangersRule::backward(const scalar* data, int batch_size) {

    forward(data, outputs_, batch_size);

    cblas_dgemm(CblasRowMajor, CblasTrans, CblasNoTrans,
                in_dims_, out_dims_, batch_size,
                1.0, data, in_dims_, outputs_, out_dims_,
                0.0, grad_, out_dims_
    );

    for(int k = 0; k < out_dims_; ++k) {

        cblas_dgemv(CblasRowMajor, CblasTrans,
                    batch_size, k + 1,
                    1.0, outputs_, out_dims_, outputs_ + k, out_dims_,
                    0.0, inference_, 1
        );

        cblas_dgemv(CblasRowMajor, CblasNoTrans,
                    in_dims_, k + 1,
                    -1.0, weights_, out_dims_, inference_, 1,
                    1.0, grad_ + k, out_dims_
        );

    }

}
```

Question 1 – Grading

For each of the four TO DOs, the points are awarded accordingly:

- Correct method gives (**1 point**).
- Operating on the correct matrices and vectors gives (**1 point**).
- Using correct indexing gives (**1 point**).
- Passing the unit test gives (**1 point**).
- Using a single function call as specified in the description gives (**1 point**).

Hint: In order to test your implementation of each TO DO individually, you can copy the master solution for the other TO DOs and then run the unit test.

Question 2a – Formulation

Hence, we can write a general linear neural network as given in equation (8) with the definition of a linear layer as given in equation (9).

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m, \mathbf{x} \mapsto \left(\phi^{m,h_L}_{\mathbf{W}_{L+1},\mathbf{b}_{L+1}} \circ \ldots \circ \phi^{h_2,h_1}_{\mathbf{W}_2,\mathbf{b}_2} \circ \phi^{h_1,n}_{\mathbf{W}_1,\mathbf{b}_1} \right)(\mathbf{x}) \tag{8}$$

$$\phi^{k,l}_{\mathbf{W},\mathbf{b}} : \mathbb{R}^l \longrightarrow \mathbb{R}^k, \mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}, \text{ with } \mathbf{W} \in \mathbb{R}^{k\times l},\ \mathbf{b} \in \mathbb{R}^k \tag{9}$$

a) **[10 points]** Prove that any linear neural network is an affine mapping.

Question 2a – Solution & Grading

1) Formulate a predicate
   2 Points

$P(k) = 1 :\Leftrightarrow$ All linear neural networks with k layers are affine transformations

2) Prove the base case
   2 Points

Using $k = 0$, we have the identity mapping, which is affine with $\boldsymbol{W} = \boldsymbol{I}$ and $\boldsymbol{b} = \boldsymbol{0}$.
Using $k = 1$, we only have one layer which is affine by definition.

3) Assume the predicate holds for some value
   1 Points

For some $k \in \mathbb{N}$, assume $P(k) = 1$

4) Induction step
   4 Points

$$
\begin{aligned}
&\left(\phi^{h_{k+1},h_k}_{\boldsymbol{W}_{k+1},\boldsymbol{b}_{k+1}} \circ \phi^{h_k,h_{k-1}}_{\boldsymbol{W}_k,\boldsymbol{b}_k} \circ \ \dots \circ \phi^{h_1,h_0}_{\boldsymbol{W}_1,\boldsymbol{b}_1}\right)(\boldsymbol{x}) \\
&= \phi^{h_{k+1},h_k}_{\boldsymbol{W}_{k+1},\boldsymbol{b}_{k+1}}\left(\left(\phi^{h_k,h_{k-1}}_{\boldsymbol{W}_k,\boldsymbol{b}_k} \circ \ \dots \circ \phi^{h_1,h_0}_{\boldsymbol{W}_1,\boldsymbol{b}_1}\right)(\boldsymbol{x})\right) \\
&= \phi^{h_{k+1},h_k}_{\boldsymbol{W}_{k+1},\boldsymbol{b}_{k+1}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \\
&= \boldsymbol{W}_{k+1}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}_{k+1} \\
&= \boldsymbol{W}_{k+1}\boldsymbol{W}\boldsymbol{x} + (\boldsymbol{W}_{k+1}\boldsymbol{b} + \boldsymbol{b}_{k+1}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}
\end{aligned}
$$

5) Conclusion
   2 Points

$\forall k \in \mathbb{N}\colon\ P(k) = 1$

Question 2b – Formulation

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ ... \\ \mathbf{x}_N^T \end{bmatrix} = \sum_{i=1}^{min\{N,n\}} \sigma_i^2 \mathbf{u}_i \mathbf{v}_i^T \tag{10}$$

The σ-values are ordered $\sigma_1^2 \geq \sigma_2^2 \geq ... \geq \sigma_{min\{N,n\}}^2 \geq 0$, and the u- and v-vectors are orthonormal, meaning that $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$ and $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$. Lastly, we define $\mathbf{V}_k := [\mathbf{v}_1 ... \mathbf{v}_k]$ with $k \leq min\{N, n\}$.

Based on these definitions prove that $(\mathbf{V}_k, \mathbf{V}_k^T)$ is an optimizer of the objective of the linear auto-encoder as given in equation (11). How does this result relate to the PCA of $\mathbf{X}$?

$$(V_k, V_k^T) \in \underset{U \in \mathbb{R}^{n \times k}, W \in \mathbb{R}^{k \times n}}{\arg\min} \sum_{i=1}^{N} \left\| \left(\phi_{\mathbf{U},\mathbf{0}}^{n,k} \circ \phi_{\mathbf{W},\mathbf{0}}^{k,n} \right) (\mathbf{x}_i) - \mathbf{x}_i \right\|^2 \tag{11}$$

Question 2a – Solution & Grading

The first step of the proof is to transform the objective into matrix form:

$$\begin{aligned}\sum_{i=1}^{N} \left\| \left(\phi_{\mathbf{U},\mathbf{0}}^{n,k} \circ \phi_{\mathbf{W},\mathbf{0}}^{k,n} \right)(\mathbf{x}_i) - \mathbf{x}_i \right\|_2^2 &= \sum_{i=1}^{N} \|\mathbf{U}\mathbf{W}\mathbf{x}_i - \mathbf{x}_i\|_2^2 \\ &= \left\| \mathbf{X}\mathbf{W}^T\mathbf{U}^T - \mathbf{X} \right\|_F^2 \\ &= \left\| \tilde{\mathbf{X}}_{\mathbf{k}} - \mathbf{X} \right\|_F^2\end{aligned}$$

Correctly reforming the objective gives (2 point).

Question 2a – Solution & Grading

Then, we compute the rank of $\tilde{\mathbf{X}}_{\mathbf{k}}$:

$$\begin{aligned} Rank(\tilde{\mathbf{X}}_{\mathbf{k}}) &= Rank(\mathbf{X}\mathbf{U}^T\mathbf{W}^T) \\ &\leq min\{Rank(X), Rank(U), Rank(W)\} \\ &\leq min\{min\{n, N\}, min\{n, k\}, min\{n, k\}\} \\ &= min\{n, k, N\} = k \end{aligned}$$

Correctly computing an upper bound on the rank gives (**1 point**).

Question 2a – Solution & Grading

$$
\begin{aligned}
\tilde{\mathbf{X}}_{\mathbf{k}} &= \mathbf{X}\mathbf{W}^T\mathbf{U}^T \\
&= \mathbf{X}\mathbf{V}_k\mathbf{V}_k^T \\
&= \left(\sum_{i=1}^{min\{N,n\}} \sigma_i^2\mathbf{u}_i\mathbf{v}_i^T\right)\mathbf{V}_k\mathbf{V}_k^T \\
&= \left(\sum_{i=1}^{min\{N,n\}} \sigma_i^2\mathbf{u}_i\left(\mathbf{v}_i^T\mathbf{V}_k\right)\right)\mathbf{V}_k^T \\
&= \left(\sum_{i=1}^{k} \sigma_i^2\mathbf{u}_i\mathbf{e}_i^T\right)\mathbf{V}_k^T \\
&= \sum_{i=1}^{k} \sigma_i^2\mathbf{u}_i\left(\mathbf{e}_i^T\mathbf{V}_k^T\right) \\
&= \sum_{i=1}^{k} \sigma_i^2\mathbf{u}_i\mathbf{v}_i^T
\end{aligned}
$$

Proving that that using $\mathbf{U} = \mathbf{V}_k$ and $\mathbf{W} = \mathbf{V}_k^T$, results in $\tilde{\mathbf{X}}_{\mathbf{k}} = \sum_{i=1}^{k}\sigma_i^2\mathbf{u}_i\mathbf{v}_i^T$, gives (4 point).

Question 2a – Solution & Grading

Hint: Use the Eckart-Young-Mirsky theorem which states that $\sum_{i=1}^{k} \sigma_i^2 \mathbf{u}_i \mathbf{v}_i^T$ is the best rank k approximation of $\mathbf{X}$.

Concluding using the Eckart-Young-Mirsky theorem gives (**5 point**).

Question 2a – Solution & Grading

There are several relations to PCA, here we point out a few:

1. One relation to PCA is that $\mathbf{v}_1, ..., \mathbf{v}_k$ are the first k principal components of the data matrix. So a solution to PCA can be directly used to construct optimal weights to the linear auto-encoder.
2. Another is that the objective in equation (14) equals the minimal reconstruction loss if we allow for arbitrary pairs of linear encoding and decoding mappings.
3. A third relation is that both can be seen as mapping higher dimensional data onto a lower dimensional space while trying to maintain as much information as possible.

Pointing out one of the relations to PCA gives (**3 point**).