AI534—Dimension reduction and Neural networks—*Solution*

1. **Dimension reduction.** One interpretation of PCA is to seek to find projection directions such that reconstruction error is minimized. Consider a set of data points $\mathbf{x}_i \in R^d, i = 1, ..., n$ that are already centered, i.e., $\sum_i \mathbf{x}_i = 0$. Let $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$ be k projection vectors such that $\mathbf{v}_t^T \mathbf{v}_t = 1$ for all $t = 1, \cdots, k$. The projection of point $\mathbf{x}_i$ is $\mathbf{y}_i = [\mathbf{v}_1^T \mathbf{x}_i, \mathbf{v}_2^T \mathbf{x}_i, ..., \mathbf{v}_k^T \mathbf{x}_i]^T \in R^k$. The reconstruction of $\mathbf{x}_i$ is expressed as $\hat{\mathbf{x}}_i = \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t$. The reconstruction error is measured as

$$\sum_{i=1}^n |\mathbf{x}_i - \hat{\mathbf{x}}_i|^2,$$

where $|\cdot|^2$ denotes squared L_2 norm.

(a) Show that minimizing this objective is equivalent to maximizing

$$\sum_{t=1}^k \mathbf{v}_t^T \Sigma \mathbf{v}_t,$$

where $\Sigma = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T$.

$$\sum_{i=1}^n |\mathbf{x}_i - \hat{\mathbf{x}}_i|^2 = \sum_{i=1}^n \left(\mathbf{x}_i - \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t\right)^T \left(\mathbf{x}_i - \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t\right)$$

$$= \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - 2\sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i + \sum_{t_1,t_2=1}^k (\mathbf{v}_{t_1}^T \mathbf{x}_i)(\mathbf{v}_{t_2}^T \mathbf{x}_i)\mathbf{v}_{t_1}^T \mathbf{v}_{t_2}\right)$$

break the last term into two

$$= \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - 2\sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i + \sum_{t_1 \neq t_2} (\mathbf{v}_{t_1}^T \mathbf{x}_i)(\mathbf{v}_{t_2}^T \mathbf{x}_i)\mathbf{v}_{t_1}^T \mathbf{v}_{t_2} + \sum_{t_1=t_2} (\mathbf{v}_{t_1}^T \mathbf{x}_i)(\mathbf{v}_{t_2}^T \mathbf{x}_i)\mathbf{v}_{t_1}^T \mathbf{v}_{t_2}\right)$$

third term goes to 0 since $\mathbf{v}_i^T \mathbf{v}_j = 0$

$$= \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - 2\sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i + \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)(\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{v}_t\right)$$

simplify last term using $\mathbf{v}_i^T \mathbf{v}_i = 1$

$$= \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - 2\sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i + \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)(\mathbf{v}_t^T \mathbf{x}_i)\right)$$

combine the last two terms

$$= \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i\right)$$

Minimizing this objective is equivalent to maximizing only the second term:

$$\sum_{i=1}^n \sum_{t=1}^k (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i = \sum_{t=1}^k \sum_{i=1}^n (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{v}_t^T \mathbf{x}_i \qquad \textit{exchanging two summations}$$

$$= \sum_{t=1}^k \sum_{i=1}^n (\mathbf{v}_t^T \mathbf{x}_i)\mathbf{x}_i^T \mathbf{v}_t \qquad \textit{because } \mathbf{v}_t^T \mathbf{x}_i = \mathbf{x}_i^T \mathbf{v}_t$$

$$= \sum_{t=1}^k \mathbf{v}_t^T \left(\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T\right) \mathbf{v}_t \qquad \textit{pull } \mathbf{v}_t \textit{ out of the inner summation}$$

$$= \sum_{t=1}^k \mathbf{v}_t^T \Sigma \mathbf{v}_t$$

(b) Show that the optimizing the objective of $\sum_{t=1}^{k} \mathbf{v}_t^T \Sigma \mathbf{v}_t$ subject to $\mathbf{v}_t^T \mathbf{v}_t = 1$ leads to the top k eigen-vectors (with largest k eigen-values) of Σ.

For constrained optimization, we form the Lagrangian:

$$\sum_{t=1}^{k} \mathbf{v}_t^T \Sigma \mathbf{v}_t - \sum_{t=1}^{k} \lambda_t (\mathbf{v}_t^T \mathbf{v}_t - 1)$$

Take gradient wrt $\mathbf{v}_t$ and set it to zero:

$$\Sigma \mathbf{v}_t - \lambda_t \mathbf{v}_t = 0$$

This suggest that the $\mathbf{v}_t$ must be an eigen-vector of Σ with corresponding eigenvalue $_t$. Under this condition, we see that the objective becomes:

$$\begin{aligned} \sum_{t=1}^{k} \mathbf{v}_t^T \Sigma \mathbf{v}_t &= \sum_{t=1}^{k} \mathbf{v}_t^T \lambda_t \mathbf{v}_t \\ &= \sum_{i=1}^{k} \lambda_t \mathbf{v}_t^T \mathbf{v}_t \\ &= \sum_{i=1}^{k} \lambda_t \end{aligned}$$

In order to maximize this objective, we will need to pick the eigenvectors of the top k eigenvalues.

(c) Given three data points, (0,0), (1,2), (-1, -2) in a 2-d space. What is the first principal component direction (please write down the actual vector)? If you use this vector to project the data points, what are their new coordinates in the new 1-d space? What is the variance of the projected data?

Answer: *One could simply noting that the three points lie on a straight line, which must be the principal component direction. The direction of the line can be described by the unit vector: $(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}})$. That will be our first principal component direction.*

One could also follow the recipe given in class. Begin by forming the covariance matrix:

$$S = \frac{1}{3}\left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \right) = \begin{pmatrix} 2/3 & 4/3 \\ 4/3 & 8/3 \end{pmatrix}$$

We now solve for the eigenvectors by solving $S - I\lambda = 0$. Using the determinant:

$$\begin{aligned} \begin{vmatrix} 2/3 - \lambda & 4/3 \\ 4/3 & 8/3 - \lambda \end{vmatrix} &= 0 \\ (2/3 - \lambda)(8/3 - \lambda) - 16/9 &= 0 \\ \lambda^2 - 10/3\lambda &= 0 \\ \lambda(\lambda - 10/3) &= 0 \\ \Rightarrow \lambda_1 = 0, \lambda_2 = 10/3 & \end{aligned}$$

To recover the first eigenvector we solve the system

$$\begin{pmatrix} 2/3 & 4/3 \\ 4/3 & 8/3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{10}{3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

which gives the solution $(1/\sqrt{5}, 2/\sqrt{5})$.

If we project the data by this vector, we get

$$\begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ 0 & 2 & -2 \end{pmatrix} = \begin{pmatrix} 0 & \sqrt{5} & -\sqrt{5} \end{pmatrix}$$

The variance of this data is

$$\frac{1}{3}\left(0^2 + \sqrt{5}^2 + (-\sqrt{5})^2\right) = \frac{10}{3}$$

Note that above we are using MLE for the variance. If one choose to use an unbiased estimator of the variance (i.e., replacing N with N-1 in the normalization term), the solution for PC does not change but the (co)variance before and after projection will change proportionally, and everything will work out in a similar way.

2. **Neural network expressiveness**
In class, we have discussed that neural network can express any arbitrary boolean functions. Please answer the following question about neural networks. You can assume a step function for the activation function.

   (a) It is impossible to implement a XOR function $y = x_1 \oplus x_2$ using a single unit (neuron). However, you can do it with a neural net. Use the smallest network you can. Draw your network and show all the weights.
   A XOR function can be represented as $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$. This can be represented using two AND units (first layer) and a OR unit (second layer) as described in class.

   (b) Explain how can we construct a neural network to implement a Naive Bayes Classifier with Boolean features.
   Without loss of generality, here we focus on a binary Naive Bayes classifier in the form that predicts $y = 1$ if $P(X, y = 1) \geq P(X, y = 0)$. To implement this, we can compute $\log P(X, y = 1) = \sum_i \log P(x_i|y = 1) + \log P(y = 1)$. Because x_i is boolean, we have $\log P(X, y = 1) = \sum_i (x_i \log P(x_i = 1|y = 1) + \log P(x_i = 0|y = 1)(1 - x_i)) + \log P(y = 1)$, which is essentially a linear function of x_i's, and the coefficients are determined by $P(x_i = 1|y = 1), P(x_i = 0|y = 1), P(y = 1)$ etc. We can compute $P(X, y = 0)$ in a similar fashion. We can then use a hard threshold function as the activation function to output 1 if $\log P(X, y = 1) - \log P(X, y = 0) \geq 0$.

   (c) Recall that the decision function of kernelized SVM can be written as:

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}))$$

   , where $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are the training examples, α_i's are the dual variables and $K(\cdot, \cdot)$ is the kernel function. Consider the following polynomial kernel:

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^p$$

   Give a neural net with a single hidden layer that represents the above SVM decision function with the polynomial kernel. Hing: for this problem, you do not need to restrict to activation functions introduced in class.
   We just need to construct one hidden node for each support vector $(\mathbf{x}_i, y_i)$ and the weights from input to the hidden node is $\mathbf{x}_i$ and the weight of the bias term is 1. The activation function of the hidden layer is $f(s) = s^p$. There is a single output node and the weight between a hidden node and the output node is simply the corresponding $y_i \alpha_i$. The output node has a step function as activation function, which outputs 1 if its input is positive and -1 if negative.