Introduction to Machine Learning

Answers to Exercise 4

Convolutional Neural Networks & Dimension Reduction

Jingtao Min

July 18, 2022

1 Clustering with k-means

(a) $k = 1$, the optimization problem is given by:

$$\min_{\boldsymbol{\mu}} \sum_{i=1}^{3} \|\mathbf{x}_i - \boldsymbol{\mu}\|_2^2 \tag{1}$$

with the closed form solution:

$$\boldsymbol{\mu} = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}_i = \left(-\frac{1}{3}, \frac{1}{3}\right) \tag{2}$$

(b) $k = 2$, supposing the cluster centroids are initialized at $\boldsymbol{\mu}_1 = (0, 0)$ and $\boldsymbol{\mu}_2 = (1, -1)$, there will be only one cluster assignment and one update that moves $\boldsymbol{\mu}_1$ to the centroid $(-1/3, 1/3)$. No new assignment or update is conducted on $\boldsymbol{\mu}_2$ since it is away from the samples.

(c) $k = 3 =$ number of samples. In this case we can assign each sample to a cluster centered at the position of itself, so that the objective goes to zero:

$$\boldsymbol{\mu}_1 = \mathbf{x}_1 = (1, 1), \quad \boldsymbol{\mu}_2 = \mathbf{x}_2 = (-1, 1), \quad \boldsymbol{\mu}_3 = \mathbf{x}_3 = (1, -1), \qquad \widehat{R}(\boldsymbol{\mu}) = \sum_{i=1}^{3} \|\mathbf{x}_i - \boldsymbol{\mu}_i\|_2^2 = 0 \tag{3}$$

2 Dimensionality reduction with PCA

Principle component analysis (PCA) can be expressed as the following optimization problem:

$$C_* = \frac{1}{n} \min_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{W}\mathbf{z}_i - \mathbf{x}_i\|_2^2 \tag{4}$$

where $\mathbf{z}_i \in \mathbb{R}^k$ and $\mathbf{x}_i \in \mathbb{R}^d$, $k \ll d$; $\mathbf{W} \in \mathbb{R}^{d\times k}$. We assume that the data points are centered, i.e. $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$.

(a) The value of $\mathrm{tr}\left(\mathbf{W}_*\mathbf{W}_*^T\right)$. The trace can be rewritten as:

$$\mathrm{tr}\left(\mathbf{W}_*\mathbf{W}_*^T\right) = \sum_{i=1}^{d}\sum_{j=1}^{k} (w_*)_{ij}(w_*)_{ij} = \sum_{j=1}^{k}\sum_{i=1}^{d} (w_*)_{ij}(w_*)_{ij} = \mathrm{tr}\left(\mathbf{W}_*^T\mathbf{W}_*\right) = \mathrm{tr}(\mathbf{I}_k) = k \tag{5}$$

(b) Given the determined weights $\mathbf{W}_*$, the embeddings can be simply obtained by a pseudoinverse:

$$\mathbf{z}_i^* = \left(\mathbf{W}_*^T\mathbf{W}_*\right)^{-1}\mathbf{W}_*^T\mathbf{x}_i = \mathbf{W}_*^\dagger \mathbf{x}_i \tag{6}$$

(c) Let $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d \geq 0$ be the eigenvalues of the empirical covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T \in \mathbb{R}^{d\times d}$. The objective is then simply the sum of the trailing eigenvalues of the empirical covariance:

$$C_* = \sum_{i=k+1}^{d} \lambda_i \tag{7}$$

(d) In standard PCA, the mapping vectors are calculated via spectral decomposition of the empirical covariance matrix:

$$\max_{\mathbf{w}_i^T \mathbf{w}_j = \delta_{ij}} \sum_{i=1}^{k} \mathbf{w}_i^T \left(\mathbf{X}\mathbf{X}^T\right) \mathbf{w}_i, \quad \mathbf{X}\mathbf{X}^T \in \mathbb{R}^{d\times d} \tag{8}$$

In the kernelized version, we assume that $\mathbf{w}_i$ should live in the span of $\mathbf{X}$, therefore the problem is converted to determining the coefficients $\boldsymbol{\alpha}_i$:

$$\max_{\boldsymbol{\alpha}_i^T \mathbf{X}^T \mathbf{X} \boldsymbol{\alpha}_j = \delta_{ij}} \sum_{i=1}^{k} \boldsymbol{\alpha}_i^T \left(\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\right) \boldsymbol{\alpha}_i, \quad \mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{n\times n} \tag{9}$$

(e) If two features are identical in the whole dataset, it is sure that the $\mathbf{X}$ is row rank deficient by at least 1, and hence the resulting empirical covariance matrix $\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{d\times d}$ is rank deficient by at least 1, i.e. $\operatorname{rank}\left(\mathbf{X}\mathbf{X}^T\right) \le d-1$. With this one can conclude that a $d-1$ dimensional embedding can be found with perfect reconstruction (zero reconstruction error, or $C_* = 0$).

3 Convolutional layers and fully connected layers

Considering a convolutional layer with input $d \times I_{\text{in}} \times I_{\text{in}}$ and output $n \times I_{\text{out}} \times I_{\text{out}}$, where d and n are number of channels for input and output, respectively. The convolution filter has dimension $m \times m$ for each channel.

(a) The (pre-activation) output is related to the input via:

$$\begin{aligned}
\mathbf{f}_i &= \sum_{k=1}^{d} \mathbf{a}_{ik} * \mathbf{x}_k \\
(\mathbf{f}_i)_{pq} &= \sum_{k=1}^{d}\sum_{s=1}^{m}\sum_{t=1}^{m} (\mathbf{a}_{ik})_{st} (\mathbf{x}_k)_{p+m-s,q+m-t} \\
f_{ipq} &= \sum_{k=1}^{d} \sum_{p'=p}^{p+m-1} \sum_{q'=q}^{q+m-1} a_{i,k,p-p'+m,q-q'+m}\, x_{kp'q'}
\end{aligned} \tag{10}$$

We can define the following order-6 tensor,

$$A_{kp'q'}^{ipq} = \begin{cases} (\mathbf{a}_{ik})_{p-p'+m,q-q'+m} & p \le p' < p+m,\ q \le q' < q+m \\ 0 & \text{else} \end{cases} \tag{11}$$

The convolutional layer can be written as:

$$f_{ipq} = \sum_{k=1}^{d} \sum_{p',q'=1}^{I_{\text{in}}} A_{kp'q'}^{ipq}\, x_{kp'q'} = \sum_{kp'q'} A_{kp'q'}^{ipq} x_{kp'q'} \tag{12}$$

The form is already that of a fully connected linear layer. To reinstate the expression to be more explicit, one can flatten the order-3 matrices f_{ipq} and $x_{kp'q'}$ into vectors, in which process the three-element index tuples are uniquely mapped to integers:

$$(i,p,q) \mapsto (i-1)I_{\text{out}}^2 + (q-1)I_{\text{out}} + p, \qquad (k,p',q') \mapsto (k-1)I_{\text{in}}^2 + (q'-1)I_{\text{in}} + p' \tag{13}$$

And the convolutional layer can be reinstated in terms of much higher-dimensional matrices and vectors:

$$\hat{\mathbf{f}} = \hat{f}_i = \sum_{j=1}^{I_{\text{in}}^2 d} \widehat{A}_{ij}\hat{x}_j = \widehat{\mathbf{A}}\hat{\mathbf{x}}, \qquad \hat{\mathbf{f}} \in \mathbb{R}^{I_{\text{out}}^2 n}, \quad \hat{\mathbf{x}} \in \mathbb{R}^{I_{\text{in}}^2 d}, \quad \widehat{\mathbf{A}} \in \mathbb{R}^{I_{\text{out}}^2 n \times I_{\text{in}}^2 d} \tag{14}$$

(b) Since convolutional layers are a special type of fully connected linear layers, their expressivity is bounded by the expressivity of fully connected layers; in other words, the family of functions $_c(\mathbf{x};\mathbf{W})$ that can be obtained from convolutional layers is a subset of those that can be expressed via fully connected layers.

(c) The specialty of convolutional layer compared to an arbitrary fully connected linear layer is its extreme sparsity. According to the original form or from Eq. 11 one can see that there are m^2nd nontrivial elements, resulting in a highly band-limited $\widehat{\mathbf{A}}$. Expressing it in fully connected linear layer would require $I_{\text{out}}^2 I_{\text{in}}^2 nd \gg m^2nd$ elements and corresponding computational complexity.

(d) Combining the previous two answers, we understand convolutional layers as a type of layer less expressive than ordinary fully connected layers, but also a type of layer whose computational cost and dimensionality is drastically reduced (marginal to be precise) compared to its fully connected counterpart.

In problems where convolutions are assumed to be sufficient to extract important features, e.g. in image processing, convolutions should suffice to extract correlations and local textures of pixels, convolutional layers provide a pratical implementation of ANN in terms of computational cost and convergence, while maintaining expressivity.

4 Linear autoencoders and PCA

An autoencoder with linear activations can be fully expressed in linear operations and matrix-vector multiplications. Denoting the encoder as $\mathbf{A}_{\text{enc}} \in \mathbb{R}^{m\times d}$ and the decoder as $\mathbf{A}_{\text{dec}} \in \mathbb{R}^{d\times m}$, the objective of autoencoder under squared loss can be expressed in the form:

$$\min_{\mathbf{A}} \sum_i \|\mathbf{x}_i - \mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}}\mathbf{x}_i\|_2^2 = \min_{\mathbf{A}} \|\mathbf{X} - \mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}}\mathbf{X}\|_F^2 \tag{15}$$

where $\|\cdot\|_F$ is the Frobenius norm. It is then apparent that the major use of the bottleneck layer is to limit the rank of the operator so that $\operatorname{rank}(\mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}}) \leq m$. Using the Eckart-Young theorem, we see that the optimal solution should be given by:

$$\mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}}\mathbf{X} = \mathbf{X}_m = \mathbf{U}\mathbf{\Sigma}_m\mathbf{V}^T \tag{16}$$

where $\mathbf{X}_m$ is the truncated-SVD version of $\mathbf{X}$ by setting $\sigma_i = 0$ $(i > m)$, and $\mathbf{\Sigma}_m = \operatorname{diag}(\sigma_1, \cdots \sigma_m, 0 \cdots 0)$. Decomposing $\mathbf{X}$ using SVD we can unravel the optimal choice of the encoder/decoder:

$$\begin{aligned} & \mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}}\mathbf{U}\mathbf{\Sigma}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}_m\mathbf{V}^T \\ \Longrightarrow\quad & \mathbf{A}_{\text{dec}}\mathbf{A}_{\text{enc}} = \mathbf{U}\left(\mathbf{\Sigma}_m\mathbf{\Sigma}^{-1}\right)\mathbf{U}^T = \mathbf{U}\begin{bmatrix}\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^T = \mathbf{U}_m\mathbf{U}_m^T \end{aligned} \tag{17}$$

where $\mathbf{U}_m = [\mathbf{u}_1|\cdots|\mathbf{u}_m] \in \mathbb{R}^{d\times m}$ contains the left singular vectors corresponding to the m leading singular values of $\mathbf{X}$. Note the dimension match between $\mathbf{A}$ and $\mathbf{U}_m$, we therefore propose that the optimal autoencoder can be constructed using the following symmetric encoder-decoder pair (although the construction is non-unique):

$$\mathbf{A}_{\text{enc}} = \mathbf{U}_m^T = \begin{bmatrix}\mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_m^T\end{bmatrix} = \mathbf{A}_{\text{dec}}^T, \qquad \mathbf{A}_{\text{dec}} = \mathbf{U}_m = \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_m\end{bmatrix} \tag{18}$$

Noticing the relation between SVD and spectral (eigenvalue) decomposition, we have

$$\mathbf{X}\mathbf{X}^T = \left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)\left(\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\right) = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \tag{19}$$

so the left singular vectors of $\mathbf{X}$ are the same as the eigenvectors of the empirical covariance $\mathbf{X}\mathbf{X}^T$, and the singular values have the same ordering as the eigenvalues. Recalling the objective and the corresponding mapping in PCA to extract an m-dimensional embedding:

$$\min_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} \sum_i \|\mathbf{x}_i - \mathbf{W}\mathbf{W}^T\mathbf{x}_i\|_2^2 = \min_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} \|\mathbf{X} - \mathbf{W}\mathbf{W}^T\mathbf{X}\|_F^2, \quad \mathbf{W}_* = \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_m\end{bmatrix} \tag{20}$$

We now see that the autoencoder with m-dimensional bottleneck is exactly the same as the m-dimensional embedding mappings via the relation:

$$\mathbf{A}_{\text{enc}} = \mathbf{W}_*^T = \begin{bmatrix}\mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_m^T\end{bmatrix}, \qquad \mathbf{A}_{\text{dec}} = \mathbf{W}_* = \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_m\end{bmatrix} \tag{21}$$

And the loss function is also exactly the same. It is clear at this stage that there are multiple optimal solutions for the autoencoder; the simplest case, for instance, would be to apply scaling to encoder and inverse the scaling in the decoder. The embeddings might change in this sense, but the optimal loss remains the same.