HOMEWORK SET 4

CSCI 5980 Think Deep Learning (Spring 2020)

Due 11:59 pm, May 14 2020

Instruction Please typeset your homework in LaTeX and submit it as a single PDF file in Canvas. No late submission will be accepted. For each problem, your should acknowledge your collaborators if any. For problems containing multiple subproblems, there are often close logic connections between the subproblems. So always remember to build on previous ones, rather than work from scratch.

Notation We will use small letters (e.g., u) for scalars, small boldface letters (e.g., $\boldsymbol{a}$) for vectors, and capital boldface letters (e.g., $\boldsymbol{A}$) for matrices. For a matrix $\boldsymbol{A}$, $\boldsymbol{a}^i$ (supscripting) means its i-th row as a *row vector*, and $\boldsymbol{a}_j$ (subscripting) means the j-the column as a column vector, and a_{ij} means its (i,j)-th element. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}^n$ is the space of n-dimensional real vectors, and similarly $\mathbb{R}^{m\times n}$ is the space of $m \times n$ real matrices. The dotted equal sign $\doteq$ means defining.

Problem 1 (Autoencoders, deep factorization, and deep sparse coding)

(a) The geometric view of PCA says that PCA tries to fit a subspace to a collection of data points. From a modeling perspective, this means PCA makes sense only when the data points lie near a subspace. For example, if $\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{z}_i + \boldsymbol{\varepsilon}_i$ for all i, where $\boldsymbol{B}$ is a basis for a subspace and $\boldsymbol{\varepsilon}_i$'s represent small-magnitude noise, trying to find a basis for the subspace spanned by $\boldsymbol{B}$ may make a lot of sense. What happens when a small fraction of the data points deviate significantly from the subspace?

To investigate this, let's generate an orthonormal subspace basis $\boldsymbol{B} \in \mathbb{R}^{200\times 20}$ and 98 random data points on the subspace as $\boldsymbol{B}\boldsymbol{z}_i$'s where each $\boldsymbol{z}_i \in \mathbb{R}^{20}$ is iid Gaussian. So this portion of data is perfectly clean. Now let's generate 2 points that are iid Gaussian in $\mathbb{R}^{100}$—these two points will be far from the subspace $\boldsymbol{B}$ almost surely and they are "outliers". Now we have 100 data points and we collect them into a data matrix $\boldsymbol{X} \in \mathbb{R}^{200\times 100}$. Please normalize the 100 data points so that they all have unit ℓ_2 norm now. *Also, do NOT perform centering to the points for the following steps.*

- Perform PCA via SVD or eigen-decomposition on $\boldsymbol{X}$ (again, no centering step) and numerically compare the subspace spanned by the top 20 singular vectors with $\boldsymbol{B}$. Remember for two subspaces spanned by two bases $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$, their distance can be measured by $\|\boldsymbol{B}_1\boldsymbol{B}_1^\dagger - \boldsymbol{B}_2\boldsymbol{B}_2^\dagger\|_F$. What do you observe? (1/12)
- Specializing the factorization formulation for PCA for our setting is

$$\min_{\boldsymbol{A}\in\mathbb{R}^{200\times 20},\boldsymbol{Z}\in\mathbb{R}^{20\times 100}} \sum_{i=1}^{100} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{z}_i\|_2^2 . \tag{1}$$

It's equivalent to the above PCA we have done of course. Now let's consider a slight modified version:

$$\min_{\boldsymbol{A}\in\mathbb{R}^{200\times 20},\boldsymbol{Z}\in\mathbb{R}^{20\times 100}} \sum_{i=1}^{100} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{z}_i\|_2 , \tag{2}$$

i.e., sum of the ℓ_2 norm, but norm squared. Numerically solve this optimization problem (choose whatever methods you're comfortable with, and auto-differentiation is also

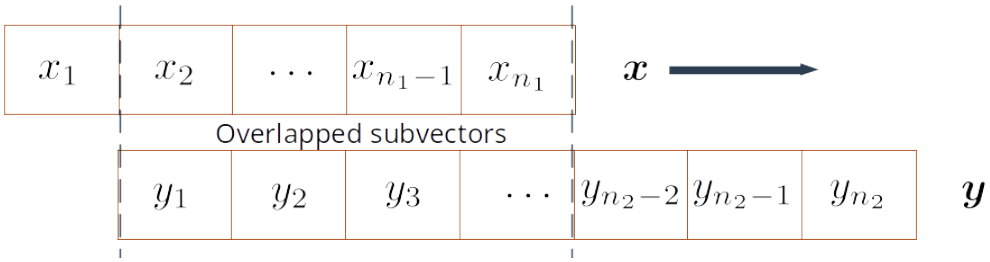
allowed: remember that PyTorch or TensorFlow can also be used to solve generic unconstrained optimization problems, not necessarily deep learning problems). Compare the subspace obtained here with $\boldsymbol{B}$. Do you get a better estimate than the plain PCA above? (2/12)

(b) Read the Science paper *Reducing the Dimensionality of Data with Neural Networks*(https://science.sciencemag.org/content/313/5786/504).

- Reproduce the first two rows of Fig 2(B), i.e., PCA and autoencoder on MNIST. You should use exactly the same architecture as provided in Fig 1 (right). The original paper uses layer-wise pretraining for initialization and conjugate-gradient for training. You probably don't need these; instead, you can choose modern initialization and optimization methods in PyTorch or TensorFlow. (2/12)
- Reproduce the plot in Fig 3, i.e., PCA and autoencoder for visualization in the two-dimensional space. The autoencoder architecture is described in the caption of Fig 3. Again, you don't need to use the original initialization and training methods. (1/12)

Problem 2 (Correlation and template matching) The word "convolutional" in convolutional neural networks is a misnomer. Cross-correlation, which is a close relative of convolution and commonly used in signal processing, is actually used. In this problem, we explore some basic properties and applications of the correlation operation. We use the standard notation $\star$ to denote cross-correlation, as against $*$ which is often used to denote convolution.

(a) For two vectors $\boldsymbol{x} \in \mathbb{R}^{n_1}, \boldsymbol{y} \in \mathbb{R}^{n_2}$, the cross-correlation $\boldsymbol{x} \star \boldsymbol{y}$ is obtained as follows:



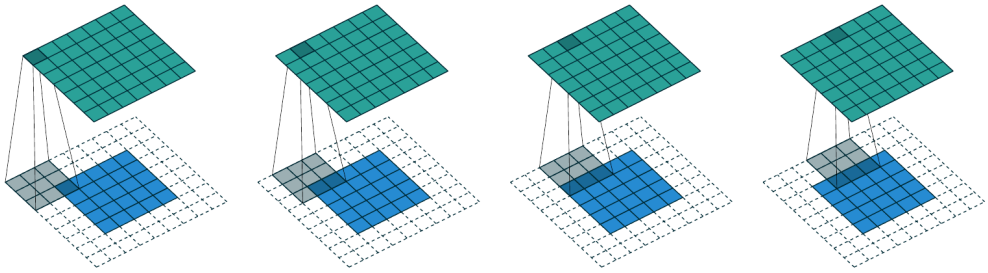
We fix the position of $\boldsymbol{y}$, and shift $\boldsymbol{x}$ to the left until $\boldsymbol{x}$ and $\boldsymbol{y}$ only have one overlapped element spatially, i.e., x_{n_1} with y_1 — that's the starting point. We calculate the inner product of two overlapped subvectors—in the beginning only two scalars actually. Then we repeatedly do this: shift $\boldsymbol{x}$ to the right by one element and calculate the corresponding inner product of the two overlapped subvectors (i.e., think of a sliding window). We end the process until $\boldsymbol{x}$ and $\boldsymbol{y}$ overlap only at one element, i.e., x_1 with y_{n_2}. The cross-correlation $\boldsymbol{x} \star \boldsymbol{y}$ is basically the vector that collects all the inner product values we have obtained in the left-to-right order. It is easy to see that $\boldsymbol{x} \star \boldsymbol{y} \in \mathbb{R}^{n_1+n_2-1}$.

Question: Calculate $[3, 2, 1] \star [4, 6, 3, 9]$. (0.5/12) For general $\boldsymbol{x}, \boldsymbol{y}$, is it true that $\boldsymbol{x} \star \boldsymbol{y} = \boldsymbol{y} \star \boldsymbol{x}$? If not, what relationship between $\boldsymbol{x} \star \boldsymbol{y}$ and $\boldsymbol{y} \star \boldsymbol{x}$ do you observe? (0.5/12)

(b) In convolutional neural networks, we have building blocks of the form $\boldsymbol{w} \star \boldsymbol{x}$, where $\boldsymbol{w}$ represents a group of learnable weights, often called *filter* following the signal processing convention. For simplicity, let's assume $\boldsymbol{w} \in \mathbb{R}^3$ and $\boldsymbol{x} \in \mathbb{R}^4$. Show that $\boldsymbol{w} \star \boldsymbol{x}$ can be written equivalently as $\boldsymbol{C}_{\boldsymbol{w}}\boldsymbol{x}$ for a certain matrix $\boldsymbol{C}_{\boldsymbol{w}} \in \mathbb{R}^{6\times 4}$ and write down $\boldsymbol{C}_{\boldsymbol{w}}$ explicitly in terms of elements of $\boldsymbol{w}$. (1/12)

(c) To apply reverse-mode auto differentiation, we need to specify $\frac{\partial}{\partial \boldsymbol{w}} (\boldsymbol{w} \star \boldsymbol{x})$, i.e., the associated Jacobian. Assume again $\boldsymbol{w} \in \mathbb{R}^3$ and $\boldsymbol{x} \in \mathbb{R}^4$, can you derive the analytic form of the Jacobian? (1/12; Hint: is it possible to write $\boldsymbol{w} \star \boldsymbol{x}$ as $\boldsymbol{C}_{\boldsymbol{x}} \boldsymbol{w}$ for a certain $\boldsymbol{C}_{\boldsymbol{x}}$?)

(d) The 2D cross-correlation is a natural generalization of the 1D cross-correlation to matrices.



(image credit: https://arxiv.org/abs/1603.07285; check out https://github.com/vdumoulin/conv_arithmetic to see the dynamic demonstration under the **Full padding, no strides** setting. Note that padding zeros, as indicated as the additional dotted boxes, is equivalent to ignoring the out-of-boundary elements.)

Compared to the 1D version, now we start from the top-left corner and end at the bottom-right corner. We scan row by row and inner products are now taken between the overlapped submatrices. All the inner product values are naturally organized into a matrix. In the pictorial illustration above, we are considering $\boldsymbol{X} \star \boldsymbol{Y}$, where $\boldsymbol{X} \in \mathbb{R}^{3\times 3}$ is the gray matrix, and $\boldsymbol{Y} \in \mathbb{R}^{5\times 5}$ is the blue matrix, the resulting green matrix $\boldsymbol{X} \star \boldsymbol{Y} \in \mathbb{R}^{7\times 7}$, where $7 = 3 + 5 - 1$.

Question: In Numpy, implement a 2D cross-correlation function. The function should take in two general matrices $\boldsymbol{Z}_1 \in \mathbb{R}^{n_1 \times n_2}$ and $\boldsymbol{Z}_2 \in \mathbb{R}^{m_1 \times m_2}$ and return the resulting cross-correlation matrix. To debug your implementation, please generate a couple of random cases and benchmark against the Scipy built-in function `scipy.signal.correlate2d` (remember to set `mode = 'full'`) https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate2d.html. (1/12)

(e) Most basic image processing algorithms are implemented as cross-correlation of a small filter $\boldsymbol{X}$ with the image of interest $\boldsymbol{Y}$. Check out the examples at the bottom of the page https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html. Use their image `ascent`, let's test your implementation of 2D cross-correlation. Try two filters

$$\boldsymbol{X}_1 = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \text{and} \quad \boldsymbol{X}_2 = \boldsymbol{X}_1^\intercal.$$

Let's say they generate two resulting matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$. Calculate $\sqrt{\boldsymbol{G}_1^2 + \boldsymbol{G}_2^2}$, where the operations are pointwise. Display your result (i.e., `imshow` as in the online example). Does your result look alike the gradient magnitude plot, except for the image boundaries? (1/12)

(f) Another way of thinking about cross-correlation is template matching. Imagine that $\boldsymbol{X}$ is a 2D pattern of interest. During the cross-correlation process, the inner product measures the agreement of the 2D pattern and local patches in $\boldsymbol{Y}$. If the value is relatively large, very likely we find a match. After we finish the cross-correlation calculation, we can spot the locations of the largest values in the cross-correlation matrix as candidate matching locations. Study the

example here https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate2d.html and compare the performance of your implementation with that of the example using `scipy.signal.correlate2d`. (1/12)