HOMEWORK SET 5

CSCI5527 Deep Learning (Spring 2026)

Due 11:59 pm, May 14 2025

Instruction Your writeup, either typeset or scanned, should be a single PDF file. For problems requiring coding, organize all codes for each top-level problem (i.e., Problem 1, Problem 2, etc) into a separate Jupyter notebook file (i.e., `.ipynb` file). Your submission to Gradescope should include the single PDF and all notebook files—**please DO NOT zip them**! No late submission will be accepted. For each problem, you should acknowledge your collaborators—**including AI tools**, if any.

About the use of AI tools You are strongly encouraged to use AI tools—they are becoming our workspace friends. It takes a bit of practice to ask the right and effective questions/prompts to these tools; we highly recommend that you go through this popular free short course **ChatGPT Prompt Engineering for Developers** offered by https://learn.deeplearning.ai/ to get started.

If you use any AI tools for any of the problems, you should include screenshots of your prompting questions and their answers in your writeup. The answers provided by such AI tools often contain factual errors and reasoning gaps. **So, if you only submit an AI answer with such bugs for any problem, you will obtain a zero score for that problem**. You obtain the scores only when you explain the bugs and also correct them in your own writing. You can also choose not to use any of these AI tools, in which case we will grade based on the efforts you have made.

Notation Please refer to our supplementary notes on high-dimensional calculus

Important notes Please provide detailed steps with justification for all problems; jumping to the final results leads to a zero score. Also, **if we ask you to use certain facts/tools to obtain something, you have to use these facts/tools (perhaps plus others); otherwise, you get a zero score**, e.g., if you prove everything from scratch when we ask you to use an existing theorem.

Problem 1 (PCA and autoencoders; 5/15)

(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 that PCA makes sense only when the data points are located near a subspace. For example, if $\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{z}_i + \boldsymbol{\varepsilon}_i$ for all i, where the columns of $\boldsymbol{B}$ form the basis for a subspace and $\boldsymbol{\varepsilon}_i$ denotes small-magnitude noise, trying to fit the data set $\{\boldsymbol{x}_i\}$ with a subspace makes a lot of sense. What happens when a small fraction of the data points deviates significantly from the subspace?

To investigate this, let's generate a random **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 the data is perfectly clean. Now let's generate 2 points that are iid Gaussian in $\mathbb{R}^{200}$—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 $\boldsymbol{X} \in \mathbb{R}^{100\times 200}$. Normalize the 100 data points so that each of them has a unit ℓ_2 norm now. *Also, do NOT perform centering to the points for the following steps.*

(i) Perform PCA via SVD on $\boldsymbol{X}$ (again, no centering step) and numerically compare the subspace spanned by the top 20 right 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 $\|\boldsymbol{B}_1\boldsymbol{B}_1^\dagger - \boldsymbol{B}_2\boldsymbol{B}_2^\dagger\|_F$, where $(\cdot)^\dagger$ denotes the pseudoinverse. What do you observe? (1/15)

(ii) Now, consider the autoencoder version of PCA, i.e.,

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

and a slightly modified version:

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

i.e., the sum of the ℓ_2 norm, not the squared norm. Numerically solve Eqs. (1) and (2) (**Note that since both problems are nonconvex, you may want to numerically solve them several times with different random initializations to pick the best solutions to amplify the chance you find global solutions**) and compare the subspace spanned by the solution $\boldsymbol{A}$ you obtain to $\boldsymbol{B}$. What do you observe? Which formulation finds a subspace closer to $\boldsymbol{B}$? (1/15)

(iii) Based on the above, design an algorithm to detect potential outliers in the given data. (1/15)

(b) Read the Science paper *Reducing the Dimensionality of Data with Neural Networks* (https://science.sciencemag.org/content/313/5786/504), which has revived deep learning since 2007.

(i) 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. (1/15)

(ii) 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 can use modern initialization and training methods instead of the original. (1/15)

Problem 2 (Self-supervised learning; 4/15)

(a) Describe the main algorithmic ideas for contrastive learning in the seminal paper https://arxiv.org/abs/2002.05709. You should assume that the reader has only a basic understanding of machine learning and deep learning. (1/15)

(b) Adapt the pretrained models from the above paper in (a) (found here https://github.com/google-research/simclr; you may want to check out their "**Model conversion to Pytorch format**" section) for image classification on CIFAR100 (https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR100.html). (1/15)

(c) Describe the main algorithmic ideas for masked autoencoders in the seminal paper https://arxiv.org/abs/2111.06377. You should assume that the reader has only a basic understanding of machine learning and deep learning. (1/15)

(d) Adapt the pretrained models from the above paper in (c) (found here https://github.com/facebookresearch/mae) for image classification on CIFAR100 (https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR100.html). (1/15)

Problem 3 (Deep generative models; $6/15$) Finally, we're here to generate new fashionable items! In other words, we will train generative models based on the famous Fashion-MNIST dataset https://github.com/zalandoresearch/fashion-mnist, which is available as a PyTorch standard dataset https://pytorch.org/docs/stable/torchvision/datasets.html#fashion-mnist.

(a) Train a GAN and generate 10 new items after training—you should display these items in your notebook. For the GAN, you can use either the original form, or any modified variation, e.g., W-GAN. $(1/15)$

(b) Modify the above implementation into a conditional GAN, i.e., with class labels as input augmentation for both generator and discriminator. Repeat training and generation and show at least 1 new item from each class. $(1/15)$

(c) Implement and train a variational autoencoder (VAE), and also generate 10 random samples from it. As is standard in VAEs, let's assume the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ and the conditional $p(\boldsymbol{x}|\boldsymbol{z})$ take multivariate Gaussian forms with diagonal covariance structures. Section 3 and Appendix C of the original paper https://arxiv.org/abs/1312.6114 can help clarify doubts. Make sure to implement the reparametrization trick so that auto-differentiation can be performed. $(2/15)$

(d) Consider vector random variables X and Z, and invertible functions f, g so that $X = f(Z)$ and $Z = g(X)$, i.e, f, g are the inverse of each other. Let $p_X(\boldsymbol{x})$ and $P_Z(\boldsymbol{z})$ denote the probability density function of X and Z, respectively. The famous multivariate change-of-variable formula reads

$$p_X(\boldsymbol{x}) = p_Z(g(\boldsymbol{x}))\left|\det \boldsymbol{J}_f(\boldsymbol{z})\right|^{-1}, \tag{3}$$

where $\boldsymbol{J}$ is our familiar notation for Jacobian. Assume that the entries inside $\boldsymbol{Z} \in \mathbb{R}^d$ are iid, and each follows a half-normal distribution (https://en.wikipedia.org/wiki/Half-normal_distribution), i.e., $\forall j \in [d]$, $Z_j \geq 0$ and $p_{Z_i}(z_i) = \frac{2}{\sqrt{2\pi}} e^{-z_j^2/2}$. Let $X = Z^2$, where the square is taken **elementwise**. Derive the probability density function for X. $(2/15)$