CS 189 Introduction to Machine Learning
Fall 2017 HW5

This homework is due **Friday, September 29 at 10pm.**

1 Getting Started

You may typeset your homework in latex or submit neatly handwritten and scanned solutions. Please make sure to start each question on a new page, as grading (with Gradescope) is much easier that way! Deliverables:

1. Submit a PDF of your writeup to assignment on Gradescope, "HW[n] Write-Up"
2. Submit all code needed to reproduce your results, "HW[n] Code".
3. Submit your test set evaluation results, "HW[n] Test Set".

After you've submitted your homework, be sure to watch out for the self-grade form.

(a) Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it:

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

2 Properties of Convex Functions

In lecture, we will introduce gradient descent as a powerful general optimization tool which, under most conditions, will converge to a local minimum of an objective function. However, a local minimum may not always be a good solution. When a function is convex, its local minima are global minima. Thus, gradient descent is an especially powerful tool for numerically optimizing convex functions. It is very useful to recognize when a function is convex or not.

Before we get to a convex function, though, let's talk about convex *sets*. A convex set is a set S where

$$x_1 \in S, x_2 \in S \implies \lambda x_1 + (1-\lambda)x_2 \in S,$$

where $0 \leq \lambda \leq 1$. In words, a convex set is a set S where x_1 in S and x_2 in S implies that everything "in between" x_1 and x_2 is also in S.

There are several equivalent ways to define a convex *function*. Here are three:

- A function f is convex if

  $$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda)x_2)$$

  for any x_1 and x_2 in the domain of f and $0 \leq \lambda \leq 1$.
- A function f is convex if the set $S_f = \{(x,y) | x \in \mathbb{R}^n, y \in \mathbb{R}, y \geq f(x)\}$ is convex. The set S_f is called the *epigraph* of the function f. It is all the points that lie "above" the curve f.
- A function f is convex if its Hessian matrix (the matrix of second partial derivatives) is positive semi-definite everywhere on the domain of f. This definition assumes the Hessian matrix exists everywhere on the domain, which is not true for non-differentiable functions.

(a) **Give an example of a function f from $\mathbb{R}$ to $\mathbb{R}$ which is convex, but not differentiable at at least one point in the domain of f.**

(b) **Compute the Hessian of the function f which maps a vector x to its loss in the ordinary least squares problem.** As a reminder, the function f is

$$f(x) = ||Ax - y||^2,$$

where $x \in \mathbb{R}^d$ and A and y are constants. This is the function we minimize to get the optimal solution x^*. **Argue that the Hessian of f is positive semi-definite.**

(c) **Prove that if f is convex and if x_0 is a local minimum of f, then x_0 is also a global minimum.** Remember that a local minimum is defined as follows: if x_0 is a local minimum, then there exists an ε such that

$$||x_0 - x|| < \varepsilon \implies f(x) \geq f(x_0).$$

(Hint: first express clearly what does it mean for a function to have a global minimum at x_0.)

(d) If f and g are convex functions, consider the functions h defined below. **Either prove that h is always convex (for any f and g) or provide a counter-example (where f and g are convex, but h is not)**:

(i) $h(x) = f(x) + g(x)$

(ii) $h(x) = \min\{f(x), g(x)\}$

(iii) $h(x) = \max\{f(x), g(x)\}$

(iv) $h(x) = f(g(x))$

3 Canonical Correlation Analysis

In this problem, we will work our way through the singular value decomposition, and show how it helps yield a solution to the problem of canonical correlation analysis.

(a) Let $n \geq d$. For a matrix $A \in \mathbb{R}^{n \times d}$ having full column rank and singular value decomposition $A = U\Sigma V^\top$, we know that the singular values are given by the diagonal entries of Σ, and the left (resp. right) singular vectors are the columns of the matrix U (resp. V). Both U and V have orthonormal columns.

Show that $A = \sum_{i=1}^{d} \sigma_i u_i v_i^\top$, where the ith singular value is denoted by $\sigma_i = \Sigma_{ii}$ and u_i and v_i are the ith left and right singular vectors, respectively.

(b) **With the setup above, show that**

i) $A^\top A$ has ith eigenvalue $\lambda_i = \sigma_i^2$, with associated eigenvector v_i.

i) $AA^\top$ has ith eigenvalue $\lambda_i = \sigma_i^2$, with associated eigenvector u_i.

Notice that both of the above matrices are symmetric.

(c) **Use the first part to show that**

$$\sigma_1(A) = \max_{\substack{u:\|u\|_2=1 \\ v:\|v\|_2=1}} u^\top A v,$$

where $\sigma_1(A)$ is the maximum singular value of A.

Additionally, show that if A has a unique maximum singular value, then the maximizers (u^*, v^*) above are given by the first left and right singular vectors, respectively.

Hint 1: You can express any $u : \|u\|_2 = 1$ as a linear combination of left singular vectors of the matrix A, and similarly with v and right singular vectors. You may or may not find this hint useful.

Hint 2: You may find the following facts that hold for any two vectors $a, b \in \mathbb{R}^d$ useful:
Cauchy-Schwarz inequality: $|a^\top b| \leq \|a\|_2 \|b\|_2$, with equality when b is a scaled version of a.
Holder's inequality: $|a^\top b| \leq \|a\|_1 \|b\|_\infty$. Here, the ℓ_1 and ℓ_∞ norms of a vector v are defined by $\|v\|_1 = \sum_i |v_i|$, and $\|v\|_\infty = \max_i |v_i|$. Let us say the vector b is fixed; then one way to achieve equality in the Holder inequality is to have:

Let i be such that $|b_i| = \|b\|_\infty$.
Set $a_i = \|a\|_1$, and $v_i = 0$ for all $j \neq i$.

(d) Define the correlation coefficient between two scalar zero-mean random variables P and Q as

$$\rho(P,Q) = \frac{\mathbb{E}[PQ]}{\sqrt{\mathbb{E}[P^2]\mathbb{E}[Q^2]}}.$$

Let us now look at the canonical correlation analysis problem, where we are given two (say, zero mean) random vectors $X, Y \in \mathbb{R}^d$, with covariance (and variance) given by

$$\begin{aligned}\mathbb{E}[XX^\top] &= \Sigma_{XX}\\ \mathbb{E}[YY^\top] &= \Sigma_{YY}\\ \mathbb{E}[XY^\top] &= \Sigma_{XY}.\end{aligned}$$

Note that a linear combination of the random variables in X can be written as $a^\top X$, and similarly, a linear combination of RVs in Y can be written as $b^\top Y$. Note that $a^\top X, b^\top Y \in \mathbb{R}$ are scalar random variables.

The goal of CCA is to find linear combinations that maximize the correlation. In other words, we want to solve the problem

$$\rho = \max_{a,b\in\mathbb{R}^d} \rho(a^\top X, b^\top Y).$$

Show that the problem can be rewritten as

$$\rho = \max_{a,b\in\mathbb{R}^d} \frac{a^\top \Sigma_{XY} b}{\left(a^\top \Sigma_{XX} a\right)^{1/2}\left(b^\top \Sigma_{YY} b\right)^{1/2}}.$$

Conclude that if (a^*, b^*) is a maximizer above, then $(\alpha a^*, \beta b^*)$ is a maximizer for any $\alpha, \beta > 0$.

(e) Assume that the covariance matrices Σ_{XX} and Σ_{YY} are full rank, and denote the maximizers in the above problem by (a^*, b^*). Use the above parts to **show that**

i) ρ^2 is the maximum eigenvalue of the matrix

$$\Sigma_{XX}^{-1/2}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{XY}^\top\Sigma_{XX}^{-1/2}.$$

ii) $\Sigma_{XX}^{1/2}a^*$ is the maximal eigenvector of the matrix

$$\Sigma_{XX}^{-1/2}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{XY}^\top\Sigma_{XX}^{-1/2}.$$

iii) $\Sigma_{YY}^{1/2}b^*$ is the maximal eigenvector of the matrix

$$\Sigma_{YY}^{-1/2}\Sigma_{XY}^\top\Sigma_{XX}^{-1}\Sigma_{XY}\Sigma_{YY}^{-1/2}.$$

Hint: An appropriate change of variables may make your life easier.

(f) **Argue why "vanilla" CCA is useless when the random vectors X and Y are uncorrelated, where by this we mean that $\mathrm{cov}(X_i, Y_j) = 0$ for all i, j.**. If you happen to know that X and Y^2 (where this is defined by squaring each entry of Y) share a linear relationship, **how might you modify the CCA procedure to account for this?**

4 Mooney Reconstruction

In this problem, we will try to restore photos of celebrities from Mooney photos, which are binarized faces. In order to do this, we will leverage a large training set of grayscale faces and Mooney faces.

Producing a face reconstruction from a binarized counterpart is a challenging high dimensional problem, but we will show that we can learn to do so from data. In particular, using the power of Canonical Correlation Analysis (CCA), we will reduce the dimensionality of the problem by projecting the data into a subspace where the images are most correlated.

Images are famously redundant and well representable in lower-dimensional subspaces as the eigenfaces example in class showed. However, here our goal is to relate two different kinds of images to each other. Let's see what happens.

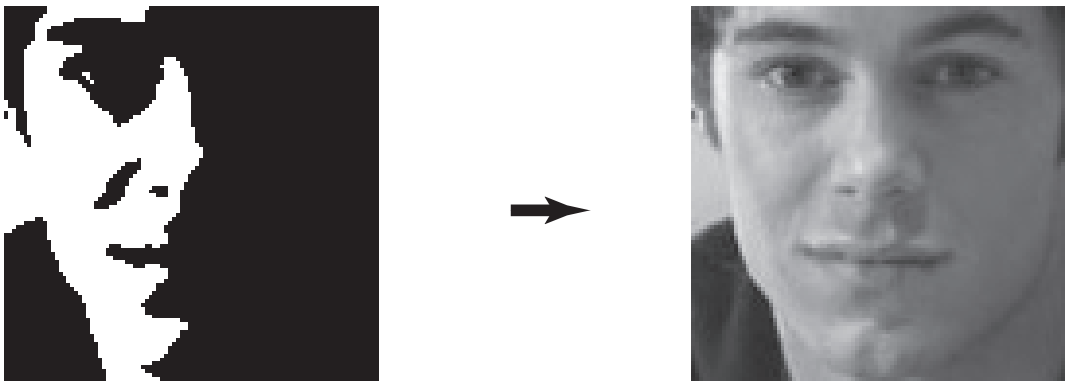

Figure 1: A
binarized Mooney image of an face being restored to its original grayscale image.

The following datasets will be used for this project: X_train.p, Y_train.p, X_test.p and Y_test.p. The training data X_train.p contains 956 binarized images, where $x_i \in \mathbb{R}^{15\times15\times3}$. The test data X_test.p contains 255 binarized images with the same dimensionality. Y_train.p contains 956 corresponding grayscale images, where $y_i \in \mathbb{R}^{15\times15\times3}$. Y_test.p contains 255 grayscale images that correspond to X_test.p.

Through out the problem we will assume that all data points are flattened and standardize as follows:

$$x = (x/255) * 2.0 - 1.0$$

(Note, however, that this standardization during loading the data does not remove the need to do actual mean removal during processing.)

Please use only the following libraries to solve the problem in Python 3.0:

```
import pickle
from scipy.linalg import eig
from scipy.linalg import sqrtm
from numpy.linalg import inv
from numpy.linalg import svd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

(a) We use CCA to find pairs of directions a and b that maximize the correlation between the projections $\tilde{x} = a^T \mathbf{x}$ and $\tilde{y} = b^T \mathbf{y}$, From the previous problem, we know that this can be done by solving the problem

$$\max \rho = \max_{a,b} \frac{a^\top \Sigma_{XY} b}{\left(a^\top \Sigma_{XX} a\right)^{1/2} \left(b^\top \Sigma_{YY} b\right)^{1/2}},$$

where Σ_{XX} and Σ_{YY} denote the covariance matrices of the X and Y images, and Σ_{XY} denotes the covariance between X and Y. Note that unlike in the previous problem, we are not given the covariance matrices and must estimate them from X, Y image samples.

Write down how to estimate the three covariance matrices from finite samples of data and implement the code for it.

(b) We know from the previous problem that we are interested in the maximum singular value of the matrix $\Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2}$, and that this corresponds to the maximum correlation coefficient ρ.

Now, however, **plot the full "spectrum" of singular values of the matrix** $(\Sigma_{XX} + \lambda I)^{-1/2} \Sigma_{XY} (\Sigma_{YY} + \lambda I)^{-1/2}$**.** Note for numerical issues we need to add a very small scalar times the identity matrix to the covariance terms. Set $\lambda = 0.00001$.

(c) You should have noticed from the previous part that we have some singular value decay. It therefore makes sense to only consider the top singular value(s), as in CCA. Let us now try to project our images x on to the subspace spanned by the top k singular values. Given that the SVD $U \Sigma V^T = \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2}$, we can use the left hand eigenvectors to form a projection.

$$P_k = \begin{bmatrix} u_0, & u_1, & \ldots & u_k \end{bmatrix},$$

Show/visualize the "face" corresponding to the first eigenvector u_0**.** . Use the following code for the visualization.

```
  def plot_image(self, vector):
    vector = ((vector+1.0)/2.0)*255.0

```

```
    vector = np.reshape(vector,(15,15,3))

    p = vector.astype("uint8")

    p = cv2.resize(p,(100,100))
    count = 0

    cv2.imwrite('eigen_face.png',p)
```

(d) We will now examine how well the projected data helps generalization when performing regression. You can think of CCA as a technique to help learn better features for the problem. We will use ridge regression regression to learn a mapping, $w \in \mathbb{R}^{k \times 675}$, from the projected binarized data to the grayscale images. The binarized images are placed in matrix $X \in \mathbb{R}^{956 \times 675}$

$$\min_w ||(XP_k)w - Y||_2^2 + \lambda ||w||_2^2$$

Implement Ridge Regression with $\lambda = 0.00001$. Plot the Squared Euclidean test error for the following values of k (the dimensions you reduce to):

$k = \{0, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 650\}$**.**

(e) **Try running the learned model on 4 of the images in the test set and report the results. Give both the binarized input, the true grayscale, and the output of your model.** Note: You can use the code from above to visualize the images.

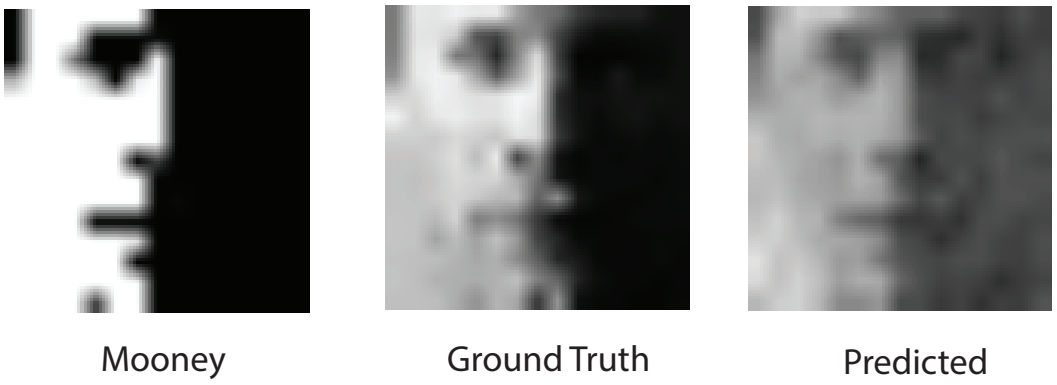

Figure 2: E
xample results with the input on the left and output on the right

5 Fruits!

The goal of the problem is help the class build a dataset for image classification, which will be used later in the course to classify fruits and vegetables. Please pick ten of the following fruits:

1. Apple

2. Banana
3. Oranges
4. Grapes
5. Strawberry
6. Peach
7. Cherry
8. Nectarine
9. Mango
10. Pear
11. Plum
12. Watermelon
13. Pineapple

Take two pictures of each specific fruit, for a total of 20 fruit pictures, against any background such that the fruit is centered in the image and the fruit takes up approximately a third of the image in area; see below for examples. Save these pictures as .png files. **Do not** save the pictures as .jpg files, since these are lower quality and will interfere with the results of your future coding project. Place all the images in a folder titled *data*. Each image should be titled *[fruit name]_[number].png* where [number]$\in \{0, 1\}$. Ex: apple_0.png, apple_1.png, banana_0.png, etc ... (the ordering is irrelevant). Please also include a file titled *rich_labels.txt* which contain entries on new lines prefixed by the file name *[fruit name]_[number]*, followed by a description of the image (maximum of eight words) with only a space delimiter in between. Ex: apple_0 one fuji red apple on wood table. To turn in the folder compress the file to a .zip and upload it to Gradescope.

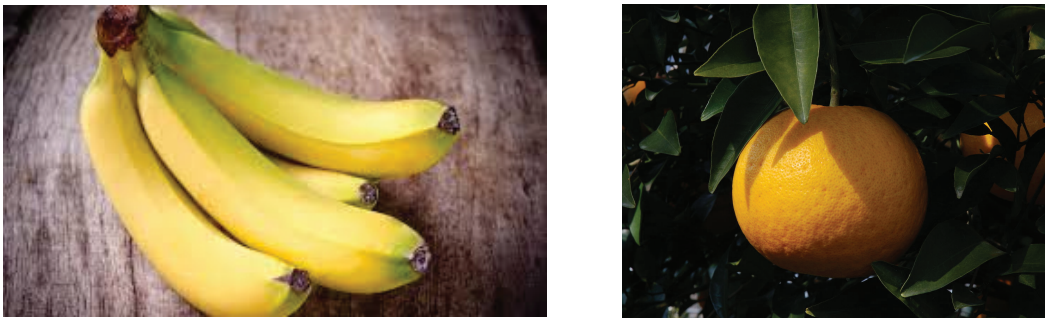

Figure 3: Example of properly centered images of four bananas against a table (left) and one orange on a tree (right).

Please keep in mind that data is an integral part of Machine Learning. A large part of research in this field relies heavily on the integrity of data in order to run algorithms. It is, therefore, vital that your data is in the proper format and is accompanied by the correct labeling not only for your grade on this section, but for the integrity of your data.

Note that if your compressed file is over 100 MB, you will need to downsample the images. You can use the following functions from skimage to help.

```
from skimage.io import imread, imsave
from skimage.transform import resize
```

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.