Introduction to Machine Learning (67577)

Exercise 7
PCA and Neural Networks

June 2018

Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to **ex_7_First_Last.zip** (First is your first name, Last is your last name. In English, of course).
- The .pdf file size should not exceed 10Mb.
- Each question is enumerated, use these numbers when answering your questions.
- You must submit your code in python files

PCA

1. Theoretical

   (a) (15 Points) Let X be a random variable with some distribution over $\mathbb{R}^d$ so that its mean is $(0, \ldots, 0) \in \mathbb{R}^d$ and its covariance matrix is $\Sigma \in \mathbb{R}^{d \times d}$. Show that for any $v \in \mathbb{R}^d$, where $\|v\|_2 = 1$, the variance of $\langle v, X \rangle$ is not larger then variance of the PCA embedding of X into 1- dimension (Assume that the PCA uses the actual Σ).

2. Coding- now we will examine a scenario in which we had n datapoints in a k-dimensional linear subspace of $\mathbb{R}^d$, where $k < d$ (e.g. in the original dataset only the first 10 coordinates can have a non zero value). As in real life, the datapoints got corrupted. We will examine a scenario in which the data was corrupted by an additive Gaussian noise.

   Before handling the noisy data, one would like to get rid of directions of the data that are a result of pure noise (e.g. coordinates $11, 12, \ldots, d$). Let's see how we can do that with PCA:

   * **Important:** We would like you to implement the original version of the PCA as shown in subsection (2.2) of recitation 13- where one needs to take the mean of the noisy data before doing any computation.

(a) (10 points) Repeat this question for $\sigma \in \{0, 0.05, 0.1, 0.15, 0.2, 0.4, 0.6\}$:

- Generate a data matrix $X \in \mathbb{R}^{n \times d}$ where $n = 100$, $d = 100$, with rank 10 so that its singular values will be $\{\sqrt{20}, \sqrt{18}, \ldots, \sqrt{2}\}$.
  * You may use an svd decomposition on some random matrix.
  * In order to make things easier for you- construct an X so that the mean of each column will be 0 (meaning that the empirical mean of the data will be 0). You may use this code:

```
import numpy as np
A= np.random.randn(n,d)
A_mean_mat= np.tile(np.mean(A,0).reshape((1,-1)),(n,1))
A -= A_mean_mat
U,_,V = np.linalg.svd(A,full_matrices= True)
D= np.sqrt(2*np.arange(10,0,-1))
X= np.dot(np.dot(U[:,:10], np.diag(D)), V[:10,:])
```

- Sample a random matrix $Z \in \mathbb{R}^{n \times d}$, where all of the matrix entries are sampled iid from the distribution $\mathcal{N}(0, \sigma^2)$.
- Generate a noisy data matrix $Y = X + Z$
- Compute the eigenvalues of noisy data's (Y) covariance matrix (Treat each row as a sample). Denote its eigen values by $\lambda_1, \ldots, \lambda_{100}$, where $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_{100}$.
- Plot (and add to your submission pdf) a bar graph of the eigenvalues in descending order - the x values will be the indexes of the eigenvalues, and the y values will be the eigenvalues values themselves, e.g. $(1, \lambda_1)$, $(2, \lambda_2), \ldots$ (you may use matplotlib.pyplot.bar)

(b) (5 points) Write in your own words the phenomena that you have experienced in (a).

(c) We will now explore the problem of choosing the embedding dimension

i. (10 Points) Given the eigenvalues of the covariance matrix- try to think of a method to distinguish between an eigenvalue that is a product of pure noise and an eigenvalue that is a product of the original data.
The only assumption that you have on your data is that originally it had a low unknown rank structure, and then it was corrupted with some noise with some unknown level.

Describe your method, even if it is a little vague, and apply it to three scenarios from question (a).

ii. (Bonus- 5 points) Calculate the inner product between the 10 leading left singular vectors of X with the 10 leading eigenvectors (the eigenvectors with the highest eigenvalues) of the covariance matrix of Y. Explain what you saw and how would you change your method based on that.

As the poet said- "You can't always get what you want".

Neural networks

Important notes before we start:

- In our next lecture- the guest lecturer will go through the two Jupyter notebooks that are supplied in the moodle site (Or an updated version of them).
- It is highly recommended to work on this exercise through Google Colab, or by using Jupyter notebook that is installed on the CS computers (Tensorboard might not be installed).
  - If you choose to work with Google Colab/Jupyter notebook - please add the file to your submission. We will not accept a link to your notebook.
- Tensorflow is a library that is written for python 3.
- If you do use Jupyter, you can change the python version by clicking on : kernel− >change kernel− > Python 3.
- If for some reason the notebook returns an error for an unknown reason, try and restarting the kernel (Kernel− > Restart).
- How to run a jupyter notebook on the CS computers:
  - Open a cmd
  - Run: module load tensorflow
  - Run: jupyter notebook mynotebook.ipynb
- One last remark- Tensorflow can define a single network at a time. Run each test seperately.

and for the fun part:

1. You can choose to run either one of the following neural nets:
   - A simple network architecture that uses only affine layers. Its located in the 12th code box in *MNIST with graphs.ipynb*
   - The Lenet Neural Network- A known neural network architecture that uses convolution layers and pooling layers. Its located in the 27-30 code boxes in *MNIST with graphs.ipynb*.
     Some more information about the convolution and pooling operations can be found in: http://cs231n.github.io/convolutional-networks/
2. Now, we are going to explore the meaning of different parameters of neural networks for each of the following items (a - e):
   - Use the given configuration while changing each time only the parameters that were requested.
   - Change the program accordingly

- Draw in a single plot the training error over the different configuration changes that you made, and in another one plot the test error (Even though the plot is in the notebook we would like you to add it to your pdf as well).
- Explain the plot that you got- do the changes result in a better neural network? Are there any concepts that we learned in the course that apply here?

(a) (15 points) GD versus SGD- We learned two algorithms: GD which uses all the training data at each step, while Stochastic GD using a batch at each iteration. Change the algorithm to run with 4 different batch sizes (by changing the command mnist.train.next_batch(32)). One of these batch sizes should be the whole training set- if your computer doesn't succeed in running it, use a big batch size instead.

In addition to the plot, state and explain how does running-time of the learning algorithm change as a function of the batch size (you can use the python package **time**).

(b) (15 points) Learning Rate - Change the algorithm to run with 3 different learning rates (by changing the command tf.train.GradientDescentOptimizer(learning_rate=0.1)). Use learning rates that are in different scales.
Try to explain your result for each learning rate, using an example of applying a gradient descent over $f(w) = w^4$ with a specific learning rate, where $f : \mathbb{R} \to \mathbb{R}$ (You may assume a specific starting point for $w_0 \in \mathbb{R}$).

(c) (10 points) Learning algorithm - Change the optimization algorithm from GradientDescentOptimizer to other optimization algorithms (e.f. to AdamOptimizer or AdagradOptimizer).

(d) (10 points) Activation Function - Change the activation function of the first hidden layer- not the one that is in the last layer (e.f., using activation_fn=tf.nn.relu).

(e) (10 points) Architecture - Change the neural network architecture by adding layers (Add affine layers with an activation layer of your choice).

Please attach all the relevant code and plots.

Bonus for life(+30% happiness in the upcoming hour): We can get pretty good results on MNIST using a simple linear classifier. A more difficult database is the CIFAR 10, which contains 10 classes of images of size 32×32. (E.g. bird, cat ,...)
You can download it from http://www.cs.toronto.edu/~kriz/cifar.html
Run the neural networks on this dataset in order to understand the learning process of the neural network.