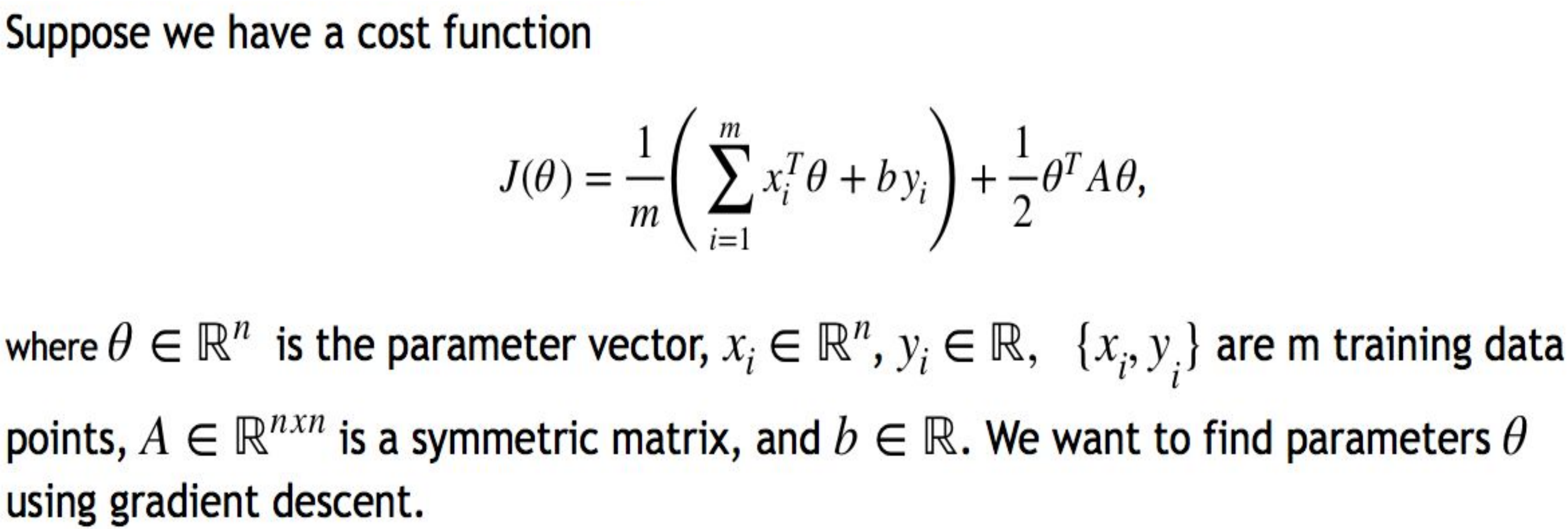
Lab 4

cs542 - Spring 2020

1. Gradient Descent

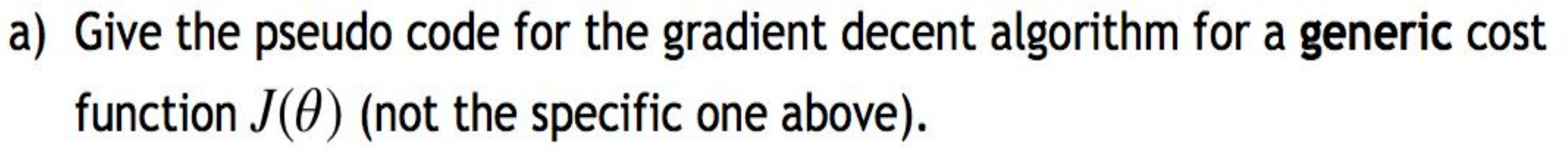


Suppose we have a cost function

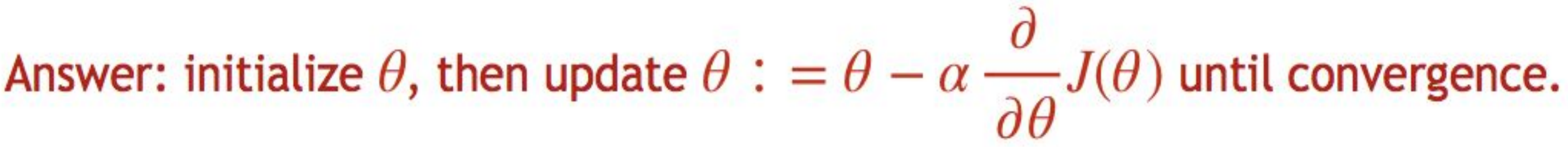
$$J(\theta) = \frac{1}{m}\left(\sum_{i=1}^{m} x_i^T\theta + by_i\right) + \frac{1}{2}\theta^T A\theta,$$

where $\theta \in \mathbb{R}^n$ is the parameter vector, $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$, $\{x_i, y_i\}$ are m training data points, $A \in \mathbb{R}^{nxn}$ is a symmetric matrix, and $b \in \mathbb{R}$. We want to find parameters θ using gradient descent.

Q1a

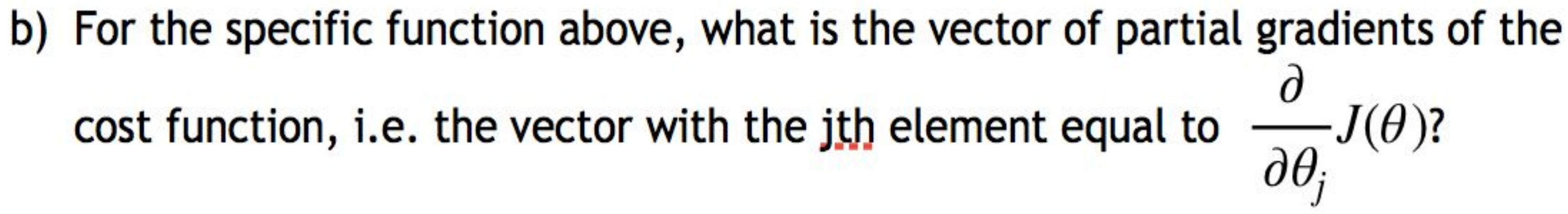


a) Give the pseudo code for the gradient decent algorithm for a **generic** cost function $J(\theta)$ (not the specific one above).



Answer: initialize θ, then update $\theta := \theta - \alpha \frac{\partial}{\partial \theta} J(\theta)$ until convergence.

Q1b



b) For the specific function above, what is the vector of partial gradients of the cost function, i.e. the vector with the jth element equal to $\frac{\partial}{\partial \theta_j} J(\theta)$?

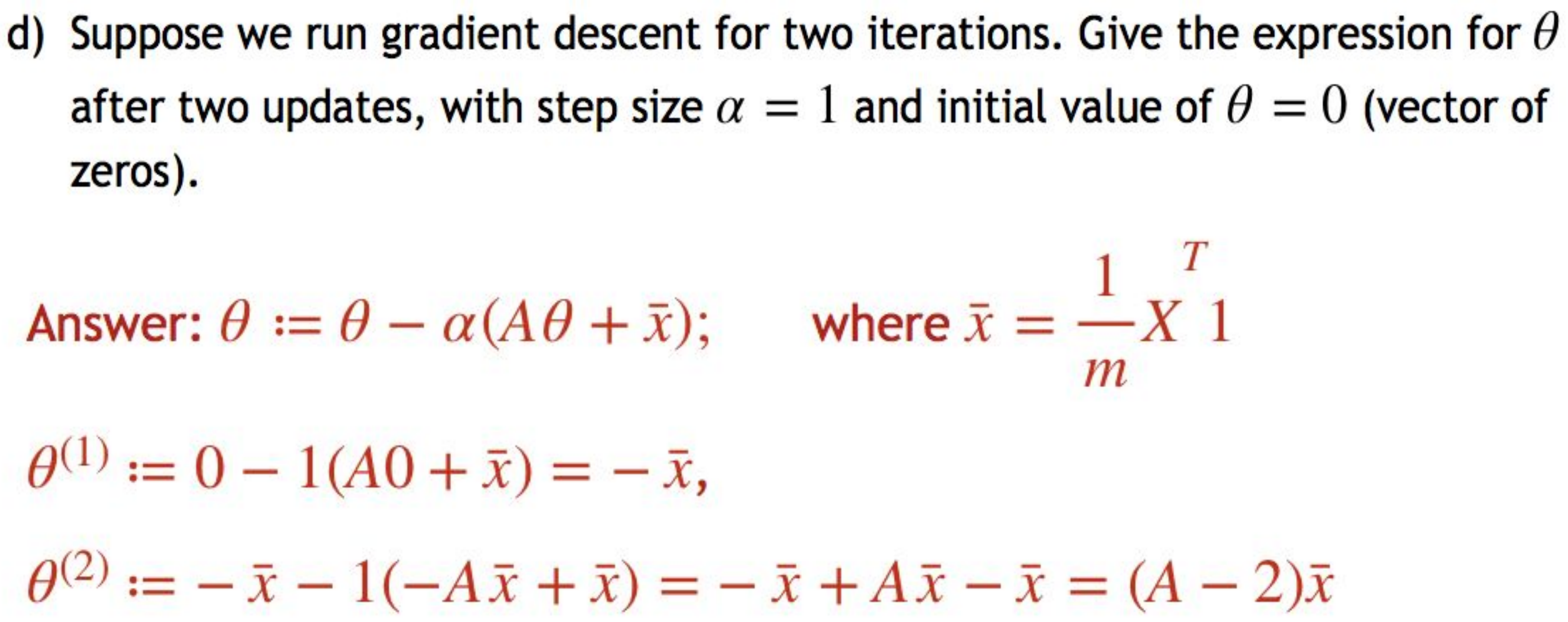
Answer: $$\frac{\partial}{\partial \theta} J(\theta) = \frac{1}{m} \sum_i x_i + \frac{1}{2} 2A\theta = \frac{1}{m} \sum_i x_i + A\theta$$

Q1c

c) Re-write the expression for the gradient without using the summation notation $\sum$. *Hint: use the design matrix X.*

Answer: $A\theta + \frac{1}{m}\sum_i x_i = A\theta + \frac{1}{m}X^T 1$, where 1 is an m-dimensional vector with all entries equal to 1. As a sanity check, the result is a nx1 vector, as it should be.

Q1d



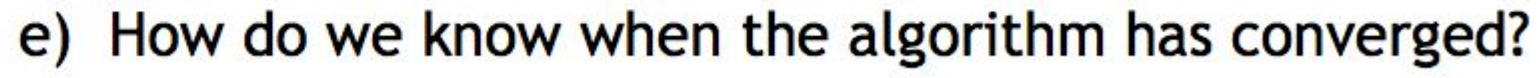
d) Suppose we run gradient descent for two iterations. Give the expression for θ after two updates, with step size $\alpha = 1$ and initial value of $\theta = 0$ (vector of zeros).

Answer: $\theta := \theta - \alpha(A\theta + \bar{x});$ where $\bar{x} = \frac{1}{m}X^{T}1$

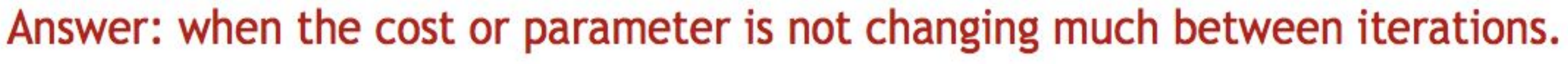
$$\theta^{(1)} := 0 - 1(A0 + \bar{x}) = -\bar{x},$$

$$\theta^{(2)} := -\bar{x} - 1(-A\bar{x} + \bar{x}) = -\bar{x} + A\bar{x} - \bar{x} = (A-2)\bar{x}$$

Q1e

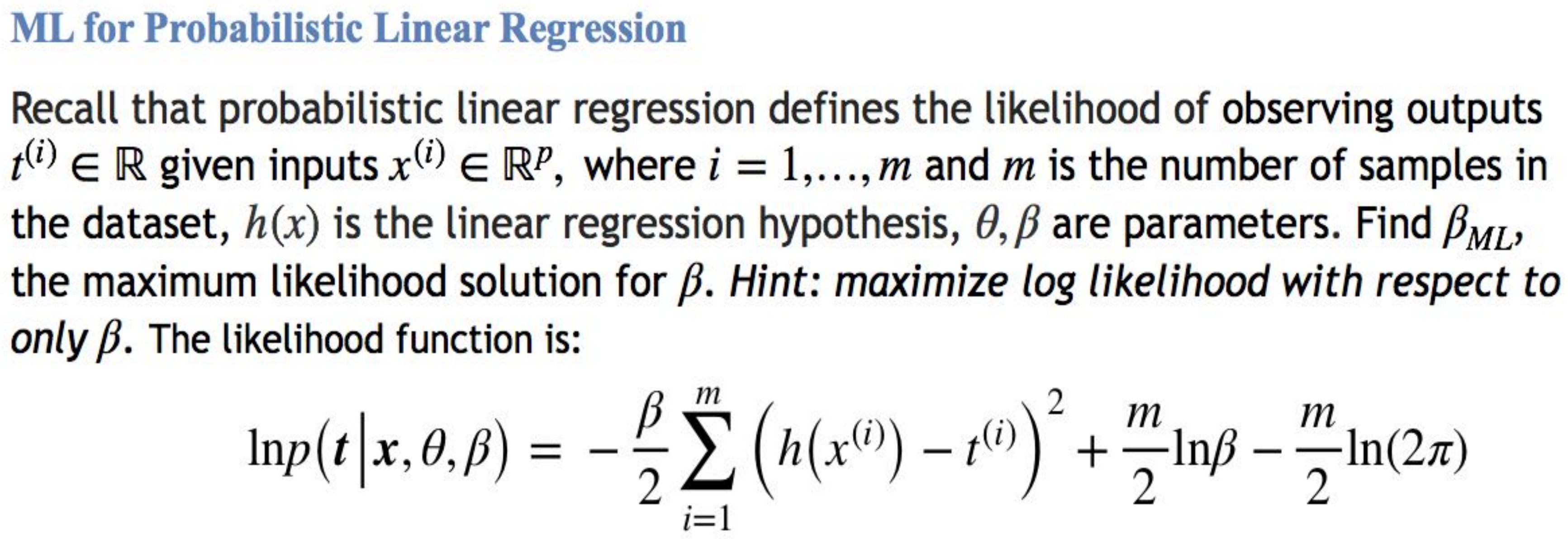


e) How do we know when the algorithm has converged?



Answer: when the cost or parameter is not changing much between iterations.

2. Maximum Likelihood Principle

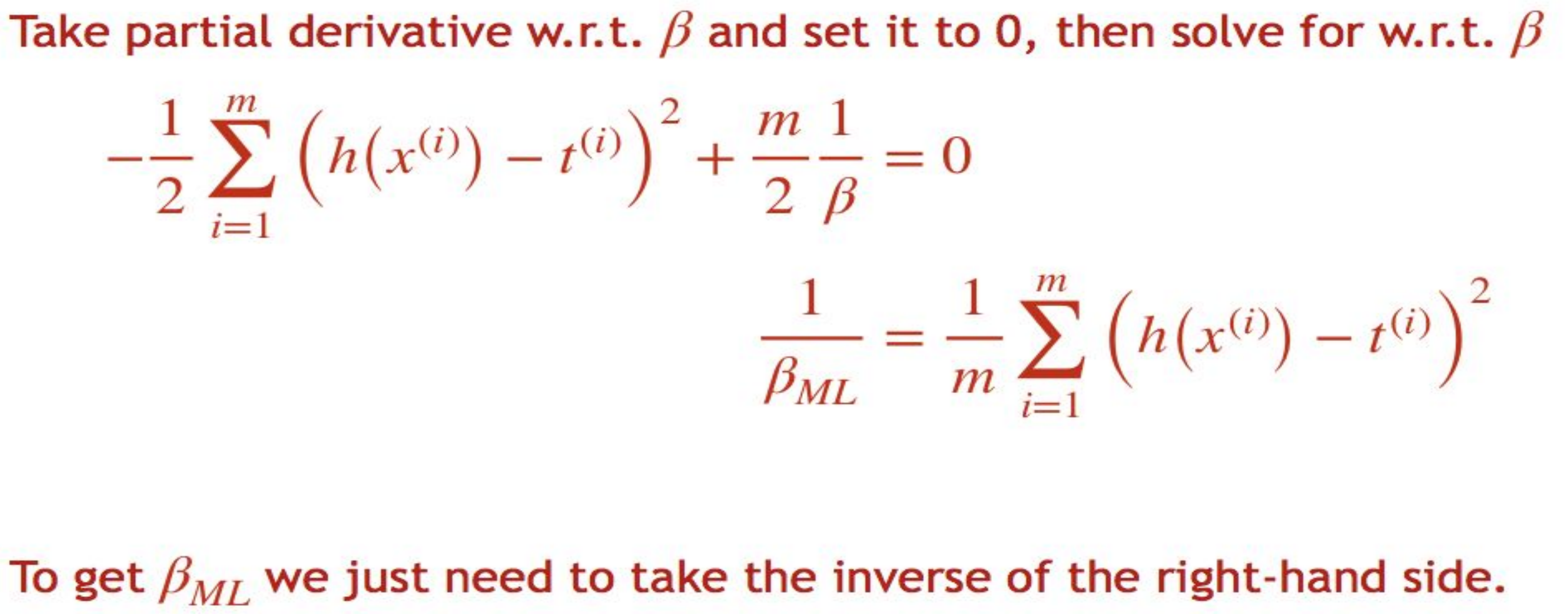


ML for Probabilistic Linear Regression

Recall that probabilistic linear regression defines the likelihood of observing outputs $t^{(i)} \in \mathbb{R}$ given inputs $x^{(i)} \in \mathbb{R}^p$, where $i = 1,\ldots,m$ and m is the number of samples in the dataset, $h(x)$ is the linear regression hypothesis, θ, β are parameters. Find β_{ML}, the maximum likelihood solution for β. *Hint: maximize log likelihood with respect to only* β. The likelihood function is:

$$\ln p\left(\boldsymbol{t} \middle| \boldsymbol{x}, \theta, \beta\right) = -\frac{\beta}{2}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - t^{(i)}\right)^2 + \frac{m}{2}\ln\beta - \frac{m}{2}\ln(2\pi)$$

Answer



Take partial derivative w.r.t. β and set it to 0, then solve for w.r.t. β

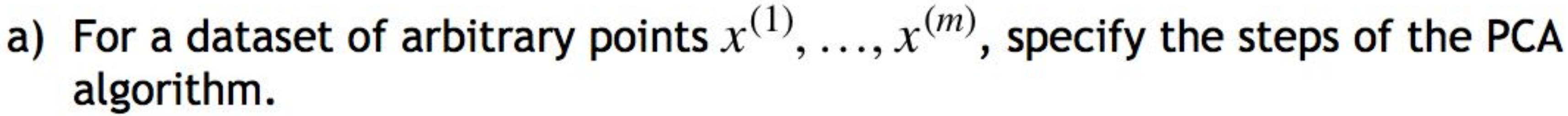
$$-\frac{1}{2}\sum_{i=1}^{m}\left(h(x^{(i)})-t^{(i)}\right)^2+\frac{m}{2}\frac{1}{\beta}=0$$

$$\frac{1}{\beta_{ML}}=\frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)})-t^{(i)}\right)^2$$

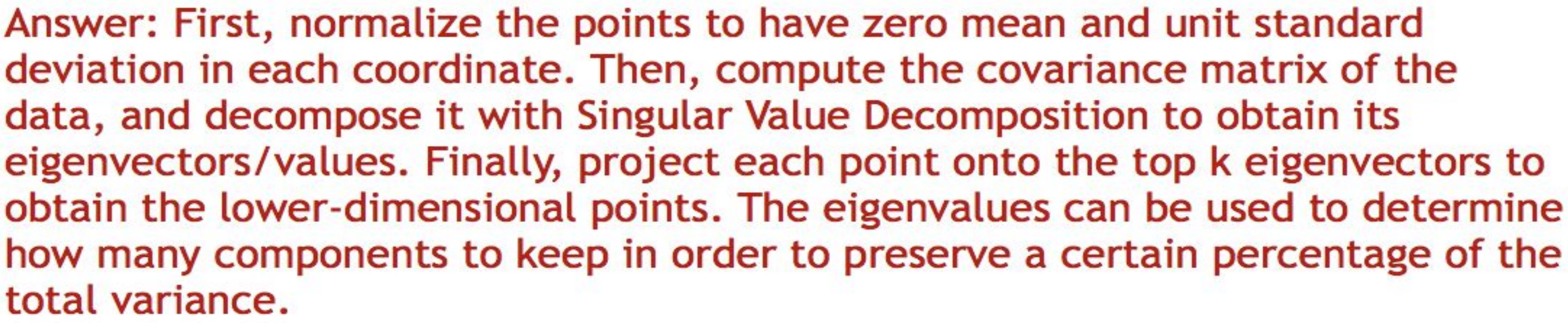
To get β_{ML} we just need to take the inverse of the right-hand side.

3. Unsupervised Learning

3.1 PCA



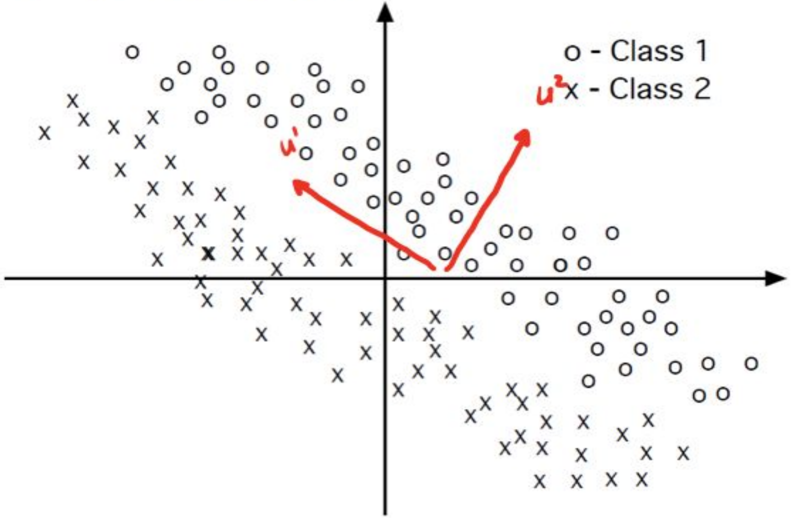
a) For a dataset of arbitrary points $x^{(1)}, \ldots, x^{(m)}$, specify the steps of the PCA algorithm.



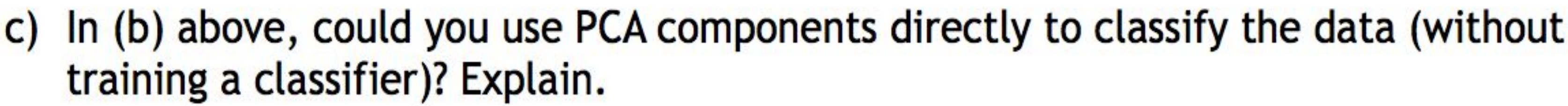
Answer: First, normalize the points to have zero mean and unit standard deviation in each coordinate. Then, compute the covariance matrix of the data, and decompose it with Singular Value Decomposition to obtain its eigenvectors/values. Finally, project each point onto the top k eigenvectors to obtain the lower-dimensional points. The eigenvalues can be used to determine how many components to keep in order to preserve a certain percentage of the total variance.

Q3.1b

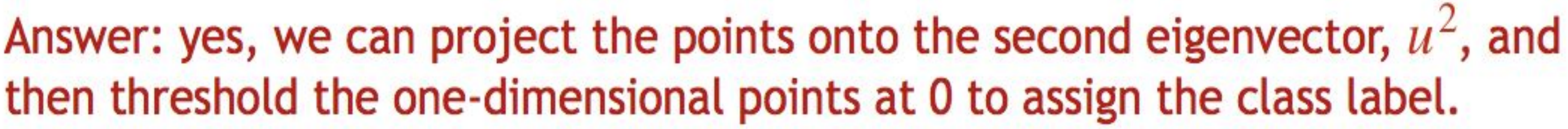
b) Suppose you are given 2D feature vectors for a classification task which are distributed according to the figure below. You apply PCA to the entire dataset. On the figure, draw all the PCA components.



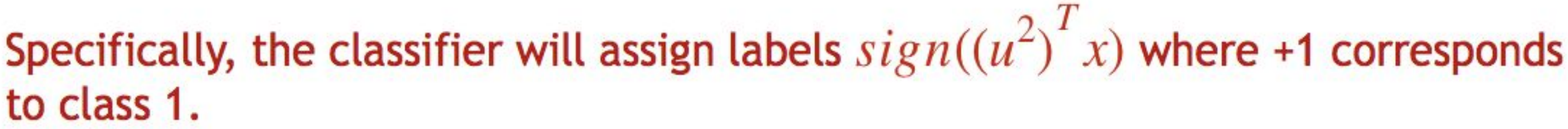
Q3.1c



c) In (b) above, could you use PCA components directly to classify the data (without training a classifier)? Explain.



Answer: yes, we can project the points onto the second eigenvector, u^2, and then threshold the one-dimensional points at 0 to assign the class label.



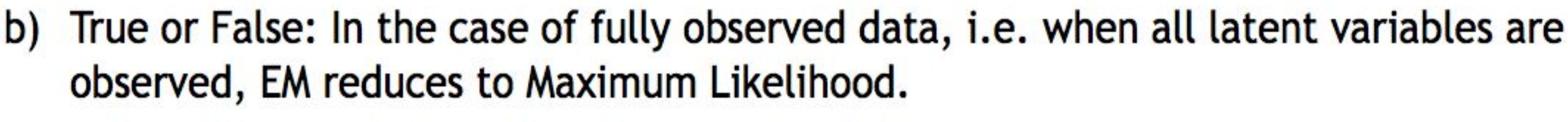
Specifically, the classifier will assign labels $sign((u^2)^T x)$ where +1 corresponds to class 1.

3.2 Gaussian Mixture Models

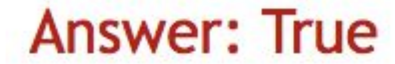
a) Describe in words the two main steps of the Expectation Maximization algorithm used to solve Gaussian Mixture Models.

Answer: the E step estimates the values of the latent variables, the M step maximizes the likelihood of the data given the latent variables computed in the E step.

Q3.2b

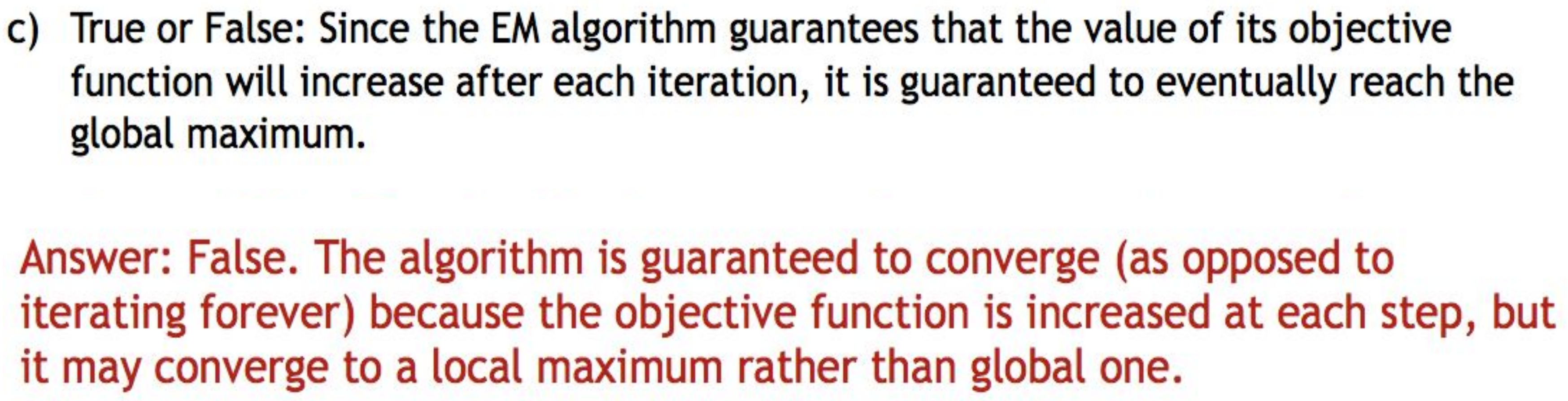


b) True or False: In the case of fully observed data, i.e. when all latent variables are observed, EM reduces to Maximum Likelihood.



Answer: True

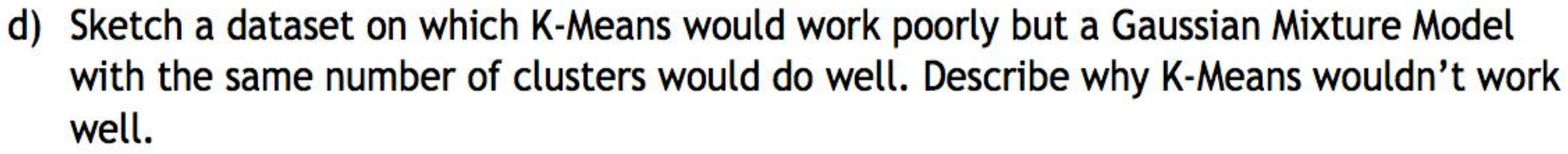
Q3.2c



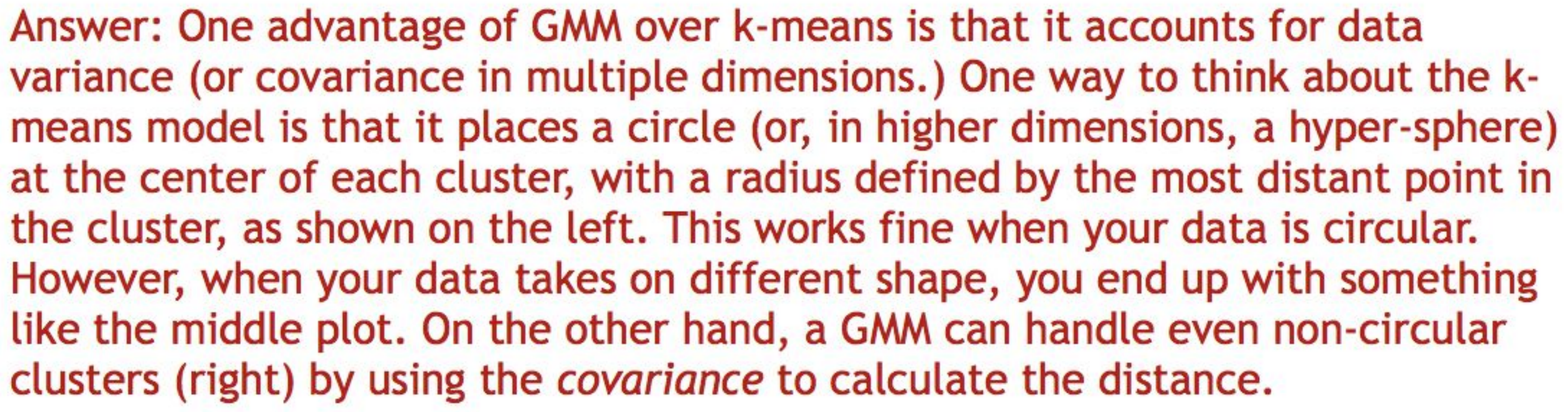
c) True or False: Since the EM algorithm guarantees that the value of its objective function will increase after each iteration, it is guaranteed to eventually reach the global maximum.

Answer: False. The algorithm is guaranteed to converge (as opposed to iterating forever) because the objective function is increased at each step, but it may converge to a local maximum rather than global one.

Q3.2d(i)

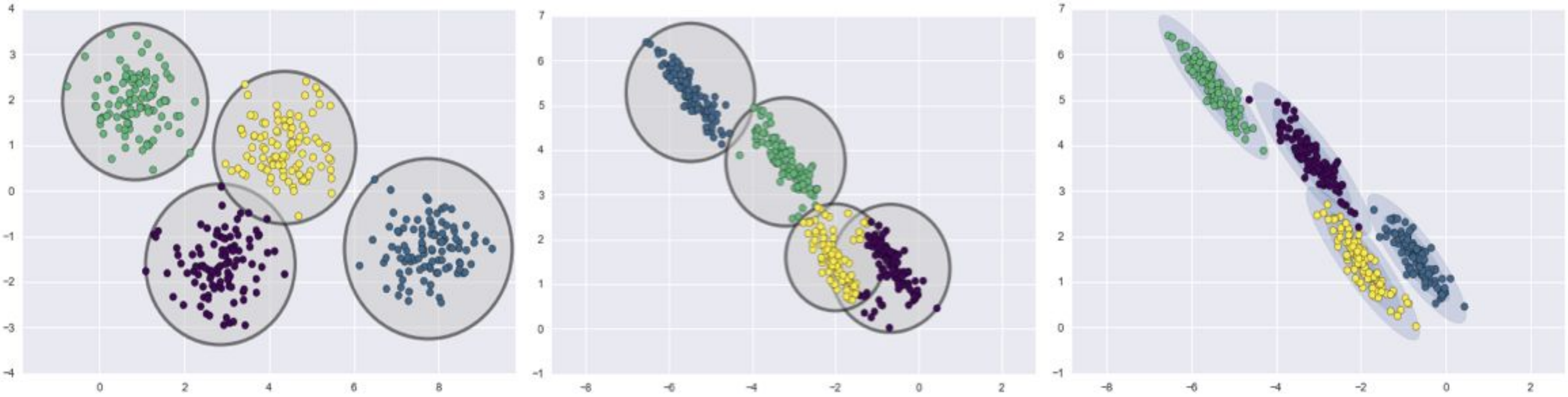


d) Sketch a dataset on which K-Means would work poorly but a Gaussian Mixture Model with the same number of clusters would do well. Describe why K-Means wouldn't work well.



Answer: One advantage of GMM over k-means is that it accounts for data variance (or covariance in multiple dimensions.) One way to think about the k-means model is that it places a circle (or, in higher dimensions, a hyper-sphere) at the center of each cluster, with a radius defined by the most distant point in the cluster, as shown on the left. This works fine when your data is circular. However, when your data takes on different shape, you end up with something like the middle plot. On the other hand, a GMM can handle even non-circular clusters (right) by using the *covariance* to calculate the distance.

Q3.2d(ii)



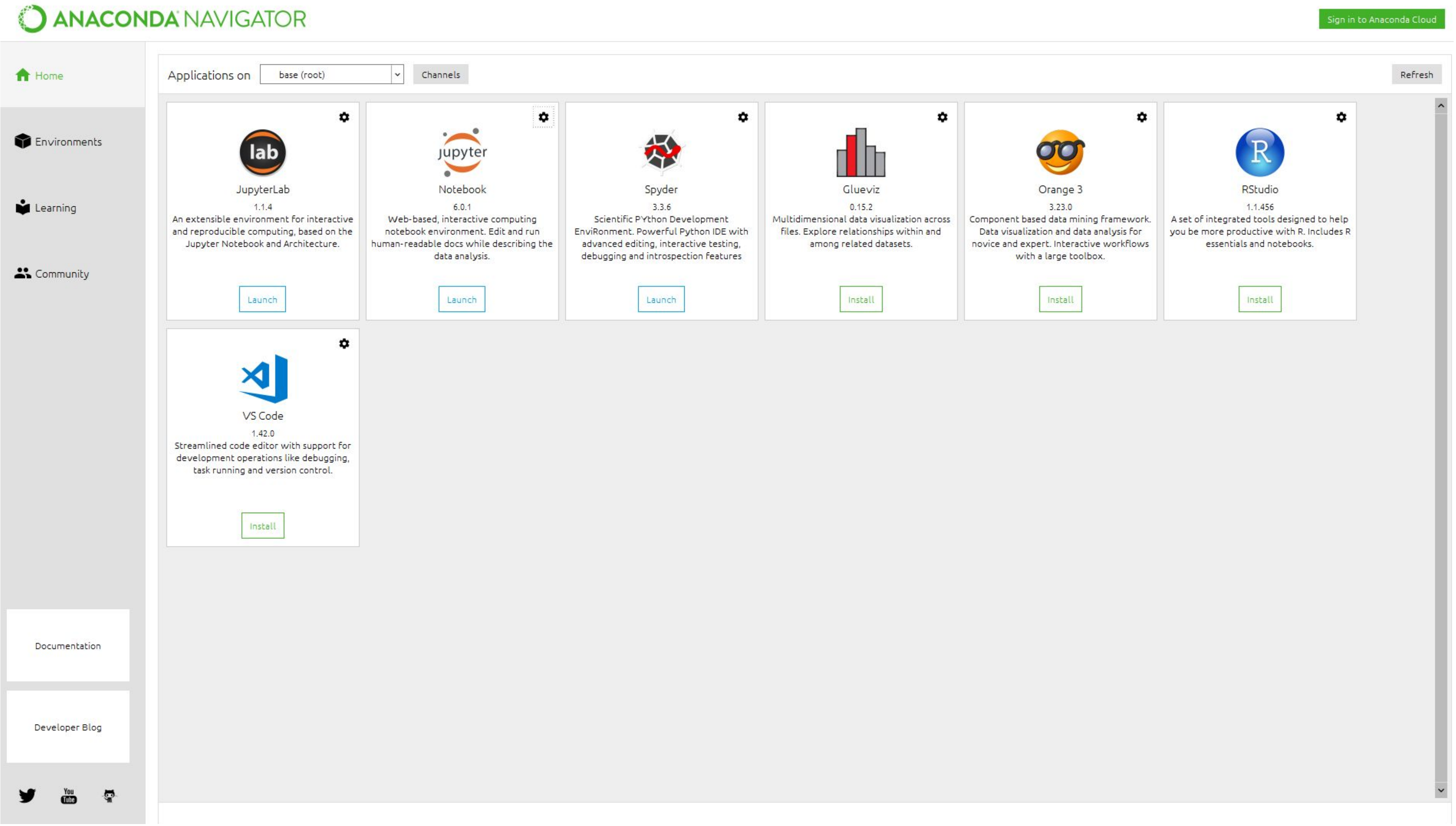
4. Anaconda Installation

- To run and solve assignments in this course, one must have a working IPython Notebook installation.

- The easiest way to set it up for both Windows and Linux is to:
    - install Anaconda: https://www.anaconda.com/distribution/ (Python Version 3)
    - save and run this file to your computer

- If you are new to Python or its scientific library, Numpy, there are some nice Tutorials: https://www.learnpython.org and http://scipy-lectures.org

- In Windows after installation, search “Anaconda Navigator”. In the GUI menu, you can launch Jupyer Notebook or Jupyter Lab. These IDEs are based on a web-browser, you can enter localhost:8888 in a web browser and enter the work directory.



JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

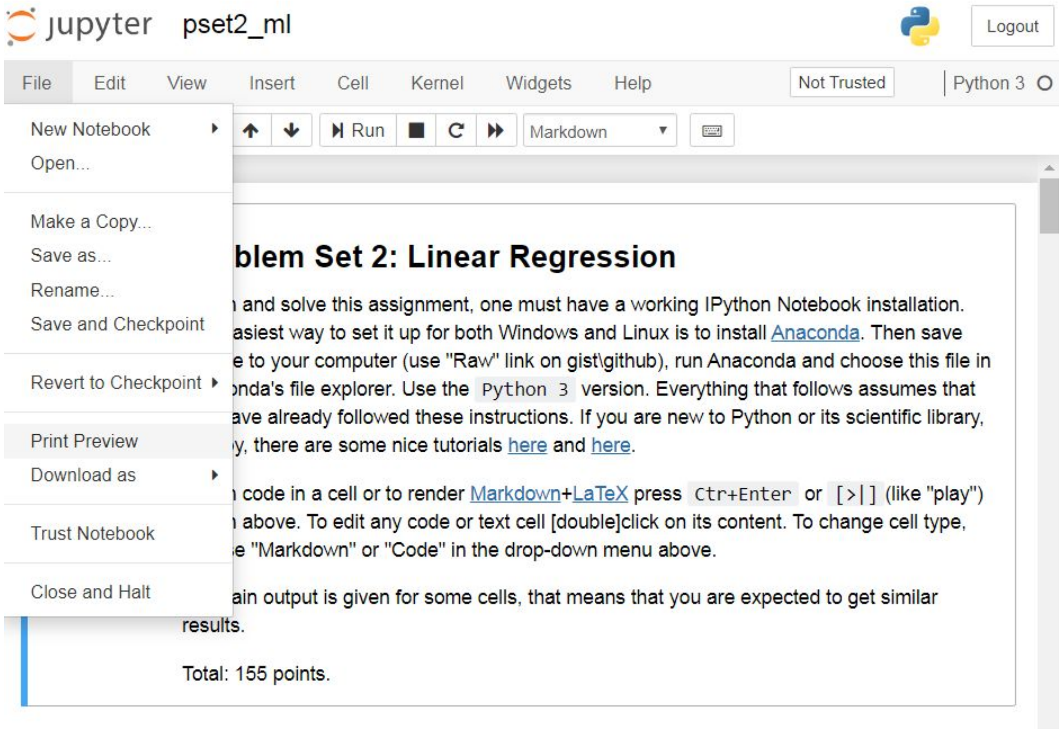
Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.23.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
1.42.0
Streamlined code editor with support for development operations like debugging, task running and version control.

5. Jupyter notebook - Save as pdf (i)



blem Set 2: Linear Regression

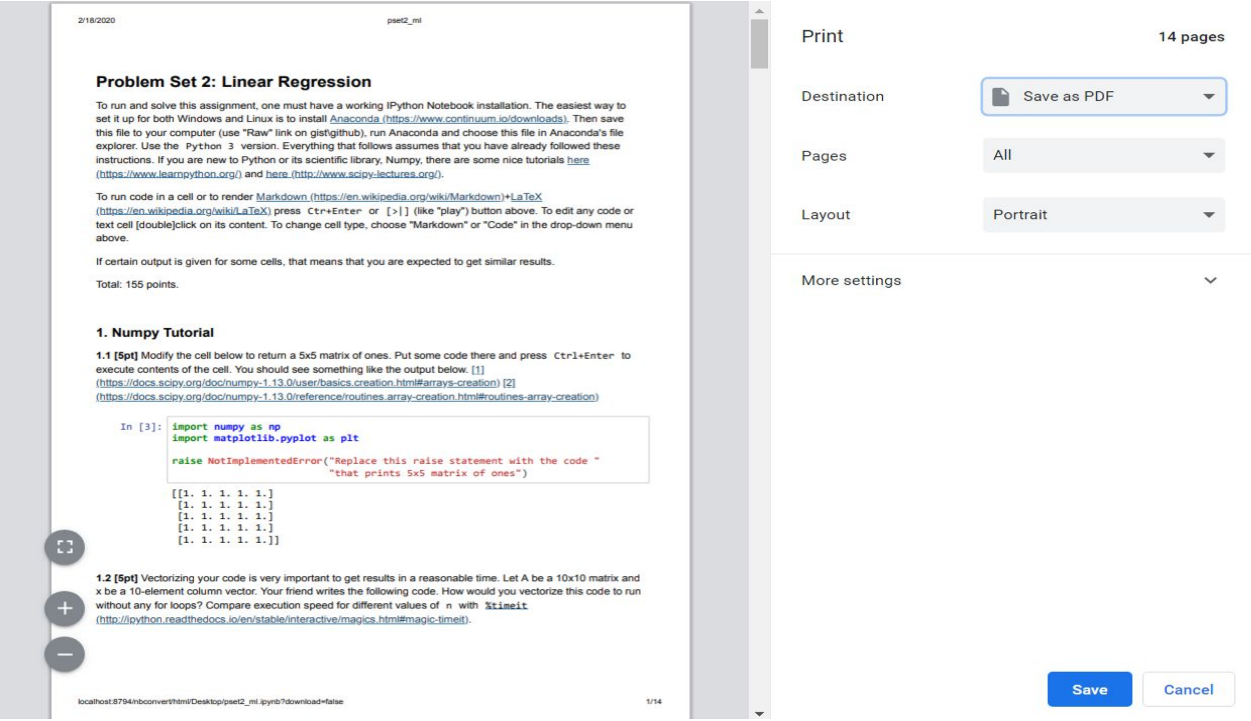
ı and solve this assignment, one must have a working IPython Notebook installation. asiest way to set it up for both Windows and Linux is to install Anaconda. Then save e to your computer (use "Raw" link on gist\github), run Anaconda and choose this file in ɔnda's file explorer. Use the `Python 3` version. Everything that follows assumes that ave already followed these instructions. If you are new to Python or its scientific library, y, there are some nice tutorials here and here.

ı code in a cell or to render Markdown+LaTeX press `Ctr+Enter` or `[>|]` (like "play") ı above. To edit any code or text cell [double]click on its content. To change cell type, e "Markdown" or "Code" in the drop-down menu above.

ain output is given for some cells, that means that you are expected to get similar results.

Total: 155 points.

Jupyter notebook - Save as pdf (ii)



2/18/2020 pset2_ml

Problem Set 2: Linear Regression

To run and solve this assignment, one must have a working IPython Notebook installation. The easiest way to set it up for both Windows and Linux is to install Anaconda (https://www.continuum.io/downloads). Then save this file to your computer (use "Raw" link on gist\github), run Anaconda and choose this file in Anaconda's file explorer. Use the `Python 3` version. Everything that follows assumes that you have already followed these instructions. If you are new to Python or its scientific library, Numpy, there are some nice tutorials here (https://www.learnpython.org/) and here (http://www.scipy-lectures.org/).

To run code in a cell or to render Markdown (https://en.wikipedia.org/wiki/Markdown)+LaTeX (https://en.wikipedia.org/wiki/LaTeX) press `Ctr+Enter` or `[>|]` (like "play") button above. To edit any code or text cell [double]click on its content. To change cell type, choose "Markdown" or "Code" in the drop-down menu above.

If certain output is given for some cells, that means that you are expected to get similar results.

Total: 155 points.

1. Numpy Tutorial

1.1 [5pt] Modify the cell below to return a 5x5 matrix of ones. Put some code there and press `Ctrl+Enter` to execute contents of the cell. You should see something like the output below. [1] (https://docs.scipy.org/doc/numpy-1.13.0/user/basics.creation.html#arrays-creation) [2] (https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.array-creation.html#routines-array-creation)

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt

        raise NotImplementedError("Replace this raise statement with the code "
                                  "that prints 5x5 matrix of ones")
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

1.2 [5pt] Vectorizing your code is very important to get results in a reasonable time. Let A be a 10x10 matrix and x be a 10-element column vector. Your friend writes the following code. How would you vectorize this code to run without any for loops? Compare execution speed for different values of `n` with `%timeit` (http://ipython.readthedocs.io/en/stable/interactive/magics.html#magic-timeit).

localhost:8794/nbconvert/html/Desktop/pset2_ml.ipynb?download=false 1/14

Print 14 pages

Destination: Save as PDF

Pages: All

Layout: Portrait