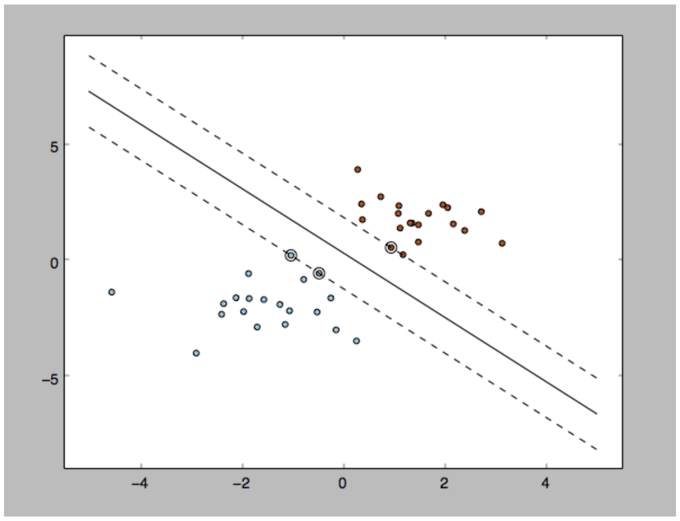
SEng 474 / CSc 578D Data Mining – Fall 2016
Assignment 3
Due: Nov 30, 2016 11:55 pm

Submit code and written answers via connex.

Different marking schemes will be used for undergrad (SEng 474) and grad (CSc 578D) students. For each question, the number of points per question for Seng 474 appears first, and CSc 578D appears second (e.g. [5, 10] -> SEng 5, CSC 10)

Always show your work. **No marks will be given for answers only.**
The coding questions use the sckit-learn, scipy and numpy libraries and should be implemented so that they run in python 2.7.

1. SVMs [20, 20]
See the code `svm_assign_q.py` adapted from
http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html
When you run the code you should get a plot that looks like this



The support vectors appear as circled dots. In the diagram above (produced with C=1 on line 11) there are three support vectors. Recall that the parameter C controls the number of non-zero slack variables.

A) Set C=10 and run the code. Are there more or less than 3 support vectors? Why did the number change/stay the same, compared to C=1 above? **[5, 5]**
B) Now set C= 0.01 and run the code. Are there more or less than 3 support vectors? Why did the number change/stay the same, compared to C=1 above? **[5, 5]**
C) In the picture generated for part B, your dashed lines will not be in the correct location. Fix the code to place the dashed lines in the correct location. Hand in a copy of the resulting graph (no code needed). **[10, 10]**

2. Clustering with k-means [15, 10]
Give an example of a dataset consisting of three natural clusters, for which (almost always) K-means would likely find the correct clusters, but bisecting K-means would not.

3. Clustering and Dendrograms [20, 20]
Use the following similarity matrix to perform single and complete link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.
("Single link clustering" means clustering using MIN, while "Complete link clustering" means clustering using MAX)

	p1	**p2**	**p3**	**p4**	**p5**
p1	0.00	0.90	0.59	0.45	0.65
p2	0.90	0.00	0.36	0.53	0.20
p3	0.59	0.36	0.00	0.56	0.15
p4	0.45	0.53	0.56	0.00	0.24
p5	0.65	0.20	0.15	0.24	0.00

4. Recommender Systems [30, 40]

In this question we will explore how to build a simple recommendation system using the jester dataset (more info on the data at http://eigentaste.berkeley.edu/dataset/). This dataset (with train/test split) is provided to you as a pickle object ('jester.p'). It is python dictionary containing the dataset, split into a training and testing set. These are accessible through the keywords 'data_train' and 'data_test'. The dataset is represented as a matrix where each row corresponds to a user and each column to a joke. The elements in the data matrix are user-joke ratings, expressed as a number between 0 and 20, where 0 means that the current joke is unrated.

Part One [30, 20]

You are asked to use the supplied python code in recommender.py, and fill in the implementation to solve the following sub-questions:

- Calculate the root mean squared error for predicting the "non-zero" values of the test data using user-user and item-item similarities (see slides) **[15, 10]**
- For each user in test set, suggest a joke that they have not rated. Make the suggestion by finding the maximum of the user-user and item-item ratings for all zero-rating jokes for the user. **[15, 10]**

Part Two: Bonus for SEng 474, required for CSc 578D [10 bonus, 20]

A. **[4 bonus, 8]** Cluster the jokes based on the correlation of user ratings and examine the text of some of the jokes in a cluster

B. **[4 bonus, 8]** Cluster the jokes based on the correlation of their actual text. To help us detect similarities even when words have suffix/prefix changes we will use character n-grams (for more info see https://en.wikipedia.org/wiki/N-gram#Examples where unit=character). We use a binary representation for the joke text, where there is a one if the word appears in the joke. Scikit-learn provides with us all the tools for making this task easy. Take a look at the documentation of "CountVectorizer" available in "sklearn.feature_extraction.text" package, and the tutorial about document clustering available at:

   http://scikit-learn.org/stable/auto_examples/text/document_clustering.html

   Please use the parameters supplied to the CountVectorizer in asn3.py for your vectorization.

C. **[2 bonus, 4]** Compare the two clustering results. How are the clusters different? Which method of clustering do you think is better and why? **Include your explanations on your paper submission handed in class.** The text of the jokes is provided you as the pickle object "jokes.pck". This is a list where each element is a joke text.

5. Gradient Descent [30, 45]

In this part of the assignment we will implement a linear regression model, using the Boston houses dataset. We will implement two techniques for minimizing the cost function: batch gradient descent, stochastic gradient descent.

You are provided with a skeleton python program and you have to implement the missing parts.

First let's recall some theory and mathematical notation [1]. We express our dataset as a matrix $X \in \mathbb{R}^{n\times m}$ where each row is a training sample and each column represent a feature, and a vector $\boldsymbol{y} \in \mathbb{R}^n$ of target values. In order to make the following notation easier, we defined a training sample as vector as $\boldsymbol{x} = [1, x_1, x_2, \ldots, x_m]$. Basically we add a column of ones to X. The purpose of linear regression is to estimate the vector of parameters $\boldsymbol{w} = [w_0, w_1, \ldots, w_m]$ such that the hyper-plane $\boldsymbol{x} \cdot \boldsymbol{w}^T$ fits, in an optimal way, our training samples. Once we have obtained $\boldsymbol{w}$ we can predict the value of testing sample, $\boldsymbol{x}_{test}$, simply by plugging it into the equation $\hat{y} = \boldsymbol{x}_{test} \cdot \boldsymbol{w}^T$. We estimate $\boldsymbol{w}$ by minimizing a cost function over the entire dataset, which is defined as follows

$$E(\boldsymbol{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{x}^{(i)} \cdot \boldsymbol{w}^T\right)^2$$

we apply gradient descent techniques to find the parameters vector $\boldsymbol{w}$ that minimizes the cost function $E(\boldsymbol{w})$. This is achieved by an iterative algorithm that starts with an initial guess for $\boldsymbol{w}$ and repeatedly performs the update

$$w_j := w_j + \kappa \frac{\partial}{\partial w_j} E(\boldsymbol{w}) \quad j = 0, 1, \ldots, m$$

where κ is the learning rate parameter.

In this question your asked to implement some gradient descent techniques. We are going to set a maximum number of iteration, `max_iter` in the code, and analyze what happens to the cost function at each iteration. Specifically you are asked to:

SEng 474; CSc 587D, 30 pts

(a) Implement the *batch gradient descent* algorithm. Plot the cost function for each iteration. Test it with different learning rate, such as κ i.e .001, .01. .1. Explain what is happening. [**15 pts**]

(b) Implement the *stochastic gradient descent* algorithm. Plot the cost function with different learning rate. Compare it with the batch gradient descent. [**15 pts**]

[1] matrix are represented by capital letter, vectors with a lower case bold letter. For convention we consider each vector as row vector (this is the opposite to what happens in standard mathematical notation).

CSc 578D only, 15 pts

(c) Now we introduce L_2 *regularization*. Explain why it is used. Implement it and repeat (b) with different values of λ i.e 10, 100, 1000. Remember that with regularization the cost function becomes

$$E(\boldsymbol{w}) = \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{x}^{(i)} \cdot \boldsymbol{w}^T\right)^2 + \lambda \sum_{j=1}^{m} w_j^2\right)$$

Note that for convention we don't regularize w_0. [**10 pts**]

(d) Propose a convergence criteria that stops the algorithm before going through all the iterations. [**5 pts**]