6.790. Identifying a system from input/output data. We consider the standard setup:
\[ y = Ax + v, \]
where \( A \in \mathbb{R}^{m \times n} \), \( x \in \mathbb{R}^n \) is the input vector, \( y \in \mathbb{R}^m \) is the output vector, and \( v \in \mathbb{R}^m \) is the noise or disturbance. We consider here the problem of estimating the matrix \( A \), given some input/output data. Specifically, we are given the following:
\[ x^{(1)}, \ldots, x^{(N)} \in \mathbb{R}^n, \quad y^{(1)}, \ldots, y^{(N)} \in \mathbb{R}^m. \]
These represent \( N \) samples or observations of the input and output, respectively, possibly corrupted by noise. In other words, we have
\[ y^{(k)} = Ax^{(k)} + v^{(k)}, \quad k = 1, \ldots, N, \]
where \( v^{(k)} \) are assumed to be small. The problem is to estimate the (coefficients of the) matrix \( A \), based on the given input/output data. You will use a least-squares criterion to form an estimate \( \hat{A} \) of \( A \). Specifically, you will choose as your estimate \( \hat{A} \) the matrix that minimizes the quantity
\[ J = \sum_{k=1}^{N} \|Ax^{(k)} - y^{(k)}\|^2 \]
over \( A \).

a) Explain how to do this. If you need to make an assumption about the input/output data to make your method work, state it clearly. You may want to use the matrices \( X \in \mathbb{R}^{n \times N} \) and \( Y \in \mathbb{R}^{m \times N} \) given by
\[
X = \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix}, \quad Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(N)} \end{bmatrix}
\]
in your solution.

b) On the course web site you will find some input/output data for an instance of this problem in the file `sysid_data.json`. Executing this Julia file will assign values to \( m \), \( n \), and \( N \), and create two matrices that contain the input and output data, respectively. The \( n \times N \) matrix variable \( X \) contains the input data \( x^{(1)}, \ldots, x^{(N)} \) (i.e., the first column of \( X \) contains \( x^{(1)} \), etc.). Similarly, the \( m \times N \) matrix \( Y \) contains the output data \( y^{(1)}, \ldots, y^{(N)} \). You must give your final estimate \( \hat{A} \), your source code, and also give an explanation of what you did.

6.810. Estimating a signal with interference. This problem concerns three proposed methods for estimating a signal, based on a measurement that is corrupted by a small noise and also by an interference, that need not be small. We have
\[ y = Ax + Bv + w, \]
where $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times p}$ are known. Here $y \in \mathbb{R}^m$ is the measurement (which is known), $x \in \mathbb{R}^n$ is the signal that we want to estimate, $v \in \mathbb{R}^p$ is the interference, and $w$ is a noise. The noise is unknown, and can be assumed to be small. The interference is unknown, but cannot be assumed to be small. You can assume that the matrices $A$ and $B$ are skinny and full rank (i.e., $m > n$, $m > p$), and that the ranges of $A$ and $B$ intersect only at 0. (If this last condition does not hold, then there is no hope of finding $x$, even when $w = 0$, since a nonzero interference can masquerade as a signal.) Each of the EE263 TAs proposes a method for estimating $x$. These methods, along with some informal justification from their proposers, are given below. Nikola proposes the **ignore and estimate method.** He describes it as follows:

We don't know the interference, so we might as well treat it as noise, and just ignore it during the estimation process. We can use the usual least-squares method, for the model $y = Ax + z$ (with $z$ a noise) to estimate $x$. (Here we have $z = Bv + w$, but that doesn't matter.)

Almir proposes the **estimate and ignore method.** He describes it as follows:

We should simultaneously estimate both the signal $x$ and the interference $v$, based on $y$, using a standard least-squares method to estimate $[x^T \ v^T]^T$ given $y$. Once we've estimated $x$ and $v$, we simply ignore our estimate of $v$, and use our estimate of $x$.

Miki proposes the **estimate and cancel method.** He describes it as follows:

Almir’s method makes sense to me, but I can improve it. We should simultaneously estimate both the signal $x$ and the interference $v$, based on $y$, using a standard least-squares method, exactly as in Almir’s method. In Almir’s method, we then throw away $\hat{v}$, our estimate of the interference, but I think we should use it. We can form the “pseudo-measurement” $\tilde{y} = y - B\hat{v}$, which is our measurement, with the effect of the estimated interference subtracted off. Then, we use standard least-squares to estimate $x$ from $\tilde{y}$, from the simple model $\tilde{y} = Ax + z$. (This is exactly as in Nikola’s method, but here we have subtracted off or cancelled the effect of the estimated interference.)

These descriptions are a little vague; part of the problem is to translate their descriptions into more precise algorithms.

a) Give an explicit formula for each of the three estimates. (That is, for each method give a formula for the estimate $\hat{x}$ in terms of $A$, $B$, $y$, and the dimensions $n, m, p$.)

b) Are the methods really different? Identify any pairs of the methods that coincide (i.e., always give exactly the same results). If they are all three the same, or all three different, say so. Justify your answer. To show two methods are the same, show that the formulas given in part (a) are equal (even if they don’t appear to be at first). To show two methods are different, give a specific numerical example in which the estimates differ.

c) Which method or methods do you think work best? Give a very brief explanation. (If your answer to part (b) is “The methods are all the same” then you can simply repeat here, “The methods are all the same.”)
6.1240. Iteratively reweighted least squares for 1-norm approximation. In an ordinary least squares problem, we are given $A \in \mathbb{R}^{m \times n}$ (skinny and full rank) and $y \in \mathbb{R}^m$, and we choose $x \in \mathbb{R}^n$ in order to minimize

$$\|Ax - y\|_2^2 = \sum_{i=1}^m (\tilde{a}_i^T x - y_i)^2.$$ 

Note that the penalty that we assign to a measurement error does not depend on the sensor from which the measurement was taken. However, this is not always the right thing to do: if we believe that one sensor is more accurate than another, we might want to assign a larger penalty to an error in the measurement from the more accurate sensor. We can account for differences in the accuracies of our sensors by assigning sensor $i$ a weight $w_i > 0$, and then minimizing

$$\sum_{i=1}^m w_i (\tilde{a}_i^T x - y_i)^2.$$ 

By giving larger weights to more accurate sensors, we can account for differences in the precision of our sensors.

a) *Weighted least squares.* Explain how to choose $x$ in order to minimize

$$\sum_{i=1}^m w_i (\tilde{a}_i^T x - y_i)^2,$$

where the weights $w_1, \ldots, w_n > 0$ are given.

b) *Iteratively reweighted least squares for $\ell_1$-norm approximation.* Consider a cost function of the form

$$\sum_{i=1}^m w_i(x)(\tilde{a}_i^T x - y_i)^2.$$  

One heuristic for minimizing a cost function of the form given in (??) is iteratively reweighted least squares, which works as follows. First, we choose an initial point $x^{(0)} \in \mathbb{R}^n$. Then, we generate a sequence of points $x^{(1)}, x^{(2)}, \ldots \in \mathbb{R}^n$ by choosing $x^{(k+1)}$ in order to minimize

$$\sum_{i=1}^m w_i(x^{(k)})(\tilde{a}_i^T x^{(k+1)} - y_i)^2.$$ 

Each step of this algorithm involves updating our weights, and solving a weighted least squares problem. Suppose we want to use this method to solve minimize the $\ell_1$-norm approximation error, which is defined to be

$$\|Ax - y\|_1 = \sum_{i=1}^m |\tilde{a}_i^T x - y_i|,$$

where the matrix $A \in \mathbb{R}^{m \times n}$ and the vector $y \in \mathbb{R}^m$ are given. How should we choose the weights $w_i(x)$ to make the cost function in (??) equal to the $\ell_1$-norm approximation error?
Numerical example. The file `11_irwls_data.json` contains data \((t_1, y_1), \ldots, (t_m, y_m)\).

We want to fit an affine model to this data:

\[
y_i = x_1 + x_2 t_i, \quad i = 1, \ldots, m.
\]

Choose \(x^{(0)}\) to be the vector of least-squares parameter estimates: that is, choose \(x^{(0)}\) in order to minimize

\[
\sum_{i=1}^{m} ((x_1^{(0)} + x_2^{(0)} t_i) - y_i)^2.
\]

Generate \(x^{(1)}, x^{(2)}, \ldots\) using iteratively reweighted least squares for \(\ell_1\)-norm approximation. You can stop generating iterates when \(\|x^{(k+1)} - x^{(k)}\| < 10^{-6}\). Report your values of \(x^{(0)}\) and the final \(x^{(k)}\) in your sequence of points. Draw a scatterplot of the data points \((t_i, y_i)\). Add the fitted lines corresponding to \(x^{(0)}\) and the final \(x^{(k)}\) to your scatterplot. What do you observe?

**Remark.** Suppose we fit the least-squares line to some data. Then, a point that is very far from the least-squares line may be an **outlier**: that is, a point that does not seem to follow the same model as the rest of the data. Because such points may not follow the same model as the rest of data, it may make sense to give such points less weight. This idea is the intuition behind iteratively reweighted least squares for \(\ell_1\)-norm approximation.