Orthogonalization problems

Coding the Matrix, 2015

`For auto-graded problems, edit the file Orthogonalization_problems.py to include your solution.`

Projection along and projection orthogonal

Ungraded problem:

1. Find the projection of $[3, 4]$ along $[5, 2]$.
2. Find the projection of $[3, 4]$ orthogonal to $[5, 2]$.
3. Find the projection of $[1, 2, 3]$ along $[-1, 1, 1]$.
4. Find the projection of $[1, 2, 3]$ orthogonal to $[-1, 1, 1]$.

Projection orthogonal to a space

Ungraded problem:

1. Find the projection of $[5, 2]$ orthogonal to the space spanned by the two vectors $[3, 4]$ and $[-7, 17]$ (which is the same as the space spanned by $[3, 4]$ and $[-12.64, 9.48]$, and the latter two vectors are orthogonal). Show each successive value of the variable `b` (as in the code for `project_orthogonal`).
2. Find the projection of $[1, 2, 3]$ orthogonal to the space spanned by the two vectors $[-1, 1, 1]$ and $[7, 5, 4]$ (which is the same as the space spanned by $[-1, 1, 1]$ and $[7.67, 4.33, 3.33]$, and the latter two vectors are orthogonal). Show each successive value of the variable `b` (as in the code for `project_orthogonal`).

Orthogonalization practice

Ungraded problem: Orthogonalize each of the following sets of vectors. Show the results of each call to `project_orthogonal`.

7. $\{[3, 4], [5, 2], [-7, 17]\}$
8. $\{[1, 2, 3], [-1, 1, 1], [7, 5, 4]\}$

Orthogonal Complement

Problem 1: Find generators for the orthogonal complement of $\mathcal{U}$ with respect to $\mathcal{W}$ where

1. $\mathcal{U} = \text{Span } \{[0, 0, 3, 2]\}$ and $\mathcal{W} = \text{Span } \{[1, 2, -3, -1], [1, 2, 0, 1], [3, 1, 0, -1], [-1, -2, 3, 1]\}$.
2. $\mathcal{U} = \text{Span } \{[3, 0, 1]\}$ and $\mathcal{W} = \text{Span } \{[1, 0, 0], [1, 0, 1]\}$.
3. $\mathcal{U} = \text{Span } \{[[-4, 3, 1, -2], [-2, 2, 3, -1]\}$ and $\mathcal{W} = \mathbb{R}^4$

Using orthogonal complement to get basis for null space

Problem 2: Let $A = \begin{bmatrix} -4 & -1 & -3 & -2 \\ 0 & 4 & 0 & -1 \end{bmatrix}$. Use orthogonal complement to find a basis for the null space of A.

Using matrix factorization to get basis for null space

Ungraded problem: Let `vlist` consist of the following vectors with domain $D = \{a, b\}$:
`Vec(D, {a:2,b:5})`, `Vec(D, {a:8,b:10})`, `Vec(D,{a:-4,b:12})`, `Vec(D,{a:1,b:-4})`. By running `orthogonalize` on this list, we get four vectors:
`Vec(D, {a:2,b:5})`, `Vec(D, {a:3.45,b:-1.38})`, `Vec(D, {})`, `Vec(D, {})`.

We write the relationship in terms of a matrix equation:

	0	1	2	3
a	2	8	-4	1
b	5	10	12	-4

[v0 | v1 | v2 | v3]

$=$

	0	1	2	3
a	2	3.45	0	0
b	5	-1.38	0	0

[v0* | v1* | v2* | v3*]

$*$

	0	1	2	3
0	1	2.28	1.79	-0.621
1	0	1	-2.2	0.65
2	0	0	1	0
3	0	0	0	1

The columns of the first matrix are the original vectors, and the columns of the second matrix are the starred vectors.

1. Give a basis for the vector space spanned by the original vectors. Explain how you got the vectors forming the basis.
2. Find the inverse of the triangular matrix. You can use your procedure from `Dimension_problems`.
3. Use the result of the previous problem to give a basis for the null space of the matrix whose columns are the original vectors.

Orthogonalization practice

Ungraded problem: Let $\boldsymbol{v}_1 = [1, 1]$ and let $\boldsymbol{v}_2 = [2, 1]$. Let `vlist` $= [\boldsymbol{v}_1, \boldsymbol{v}_2]$.

(a) Show how `orthogonalize(vlist)` computes $[\boldsymbol{v}_1^*, \boldsymbol{v}_2^*]$.

(b) Show the result of normalizing the vectors of $[\boldsymbol{v}_1^*, \boldsymbol{v}_2^*]$.

Orthonormalization

Problem 3: Write a procedure
`orthonormalize(L)` with the following spec:

- *input:* a list L of linearly independent Vecs
- *output:* a list L^* of len(L) orthonormal Vecs such that, for $i = 1, \ldots,$ len(L), the first i Vecs of L^* and the first i Vecs of L span the same space.

Your procedure should follow this outline:

1. Call `orthogonalize(L)`,
2. Compute the list of norms of the resulting vectors, and
3. Return the list resulting from normalizing each of the vectors resulting from Step 1.

Be sure to test your procedure.

When the input consists of the list of Vecs corresponding to $[4,3,1,2], [8,9,-5,-5]$, $[10,1,-1,5]$, your procedure should return the list of vecs corresponding approximately to $[0.73, 0.55, 0.18, 0.37], [0.19, 0.40, -0.57, -0.69], [0.53, -0.65, -0.51, 0.18]$.

Augmented orthonormalization

Problem 4: Write a procedure `aug_orthonormalize(L)` with the following spec:

- *input:* a list L of Vecs
- *output:* a pair `Qlist`, `Rlist` of lists of Vecs such that
  - `coldict2mat(L)` equals `coldict2mat(Qlist)` times `coldict2mat(Rlist)`, and
  - `Qlist` = `orthonormalize(L)`

Your procedure should start by calling the procedure `aug_orthogonalize(L)` defined in the module `orthogonalization`. Your procedure should *not* call `orthonormalize` but should instead incorporate code from the body of that procedure into `aug_orthonormalize`.

Here is an example for testing `aug_orthonormalize(L)`:

```
>>> L = [list2vec(v) for v in [[4,3,1,2],[8,9,-5,-5],[10,1,-1,5]]]
>>> print(coldict2mat(L))

      0  1  2
     ---------
 0  |  4  8 10
 1  |  3  9  1
 2  |  1 -5 -1
 3  |  2 -5  5

>>> Qlist, Rlist = aug_orthonormalize(L)
>>> print(coldict2mat(Qlist))

           0      1      2
     ----------------------
 0  |   0.73  0.187  0.528
 1  |  0.548  0.403 -0.653
 2  |  0.183 -0.566 -0.512
 3  |  0.365 -0.695  0.181

>>> print(coldict2mat(Rlist))

          0    1      2
     -----------------
 0  |  5.48 8.03   9.49
```

```
 1  |     0 11.4 -0.636
 2  |     0    0   6.04

>>> print(coldict2mat(Qlist)*coldict2mat(Rlist))

       0  1  2
     ---------
 0  |  4  8 10
 1  |  3  9  1
 2  |  1 -5 -1
 3  |  2 -5  5
```

Keep in mind, however, that numerical calculations are approximate:

```
>>> print(coldict2mat(Qlist)*coldict2mat(Rlist)-coldict2mat(L))

                0 1        2
     ----------------------
 0  |  -4.44E-16 0        0
 1  |           0 0 4.44E-16
 2  |  -1.11E-16 0        0
 3  |  -2.22E-16 0        0
```

QR factorization practice

Problem 5: Compute the `QR` factorization for the following matrices. You can use a calculator or computer for the arithmetic.

1. $\begin{bmatrix} 6 & 6 \\ 2 & 0 \\ 3 & 3 \end{bmatrix}$
2. $\begin{bmatrix} 2 & 3 \\ 2 & 1 \\ 1 & 1 \end{bmatrix}$

Solving a matrix-vector equation with QR factorization

Problem 6: Write and test a procedure `QR_solve(A, b)`. Assuming the columns of A are linearly independent, this procedure should return the vector $\hat{\boldsymbol{x}}$ that minimizes $\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\|$.

The procedure should use

- `triangular_solve(rowlist, label_list, b)` defined in the module `triangular`, and
- the procedure `QR_factor(A)`, defined in the stencil, which in turn uses the procedure `aug_orthonormalize(L)` that you wrote in Problem 3. Note that `QR_factor(A)` also uses `dict2list` and `list2dict`, procedures you wrote in `python_lab`.

Note that `triangular_solve` requires its matrix to be represented as a list of rows. The row-labels of the matrix R returned by `QR_factor(R)` are 0,1,2,... so it suffices to use the dictionary returned by `mat2rowdict(R)`.

Note also that `triangular_solve` must be supplied with a list `label_list` of column-labels in order

that it know how to interpret the vectors in `rowlist` as forming a triangular system. The column-labels of R are, of course, the column-labels of A. The ordering to provide here must match the ordering used in `QR_factor(A)`, which is `sorted(A.D[1], key=repr)`.

You can try your procedure on the examples given in Problem 7 and on the following example:

```
>>> A=Mat(({'a','b','c'},{'A','B'}), {('a','A'):-1, ('a','B'):2,
            ('b','A'):5, ('b','B'):3,('c','A'):1, ('c','B'):-2})
>>> print(A)

        A  B
     -------
 a  |  -1  2
 b  |   5  3
 c  |   1 -2

>>> Q, R = QR_factor(A)

>>> print(Q)

            0     1
     --------------
 a  |  -0.192  0.68
 b  |   0.962 0.272
 c  |   0.192 -0.68

>>> print(R)

         A    B
     ----------
 0  |  5.2 2.12
 1  |    0 3.54

>>> b = Vec({'a','b','c'}, {'a':1,'b':-1})
>>> x = QR_solve(A,b)
>>> x
Vec({'A', 'B'},{'A': -0.269..., 'B': 0.115...})
```

A good way to test your solution is to verify that the residual is (approximately) orthogonal to the columns of A:

```
>>> A.transpose()*(b-A*x)
Vec({'A', 'B'},{'A': -2.22e-16, 'B': 4.44e-16})
```

Least squares

Problem 7: In each of the following parts, you are given a matrix A and a vector $\boldsymbol{b}$. You are also given the approximate QR factorization of A. You are to

- find a vector $\hat{\boldsymbol{x}}$ that minimizes $||A\hat{\boldsymbol{x}} - \boldsymbol{b}||^2$,
- prove to yourself that the columns of A are (approximately) orthogonal to the residual $\boldsymbol{b} - A\hat{\boldsymbol{x}}$ by

computing the inner products, and

- calculate the value of $\|A\hat{\boldsymbol{x}} - \boldsymbol{b}\|$.

1. $A = \begin{bmatrix} 8 & 1 \\ 6 & 2 \\ 0 & 6 \end{bmatrix}$ and $\boldsymbol{b} = [10, 8, 6]$

$$A = \underbrace{\begin{bmatrix} 0.8 & -0.099 \\ 0.6 & 0.132 \\ 0 & 0.986 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 10 & 2 \\ 0 & 6.08 \end{bmatrix}}_{R}$$

2. $A = \begin{bmatrix} 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix}$ and $\boldsymbol{b} = [10, 13, 15]$

$$A = \underbrace{\begin{bmatrix} 0.424 & .808 \\ 0.566 & 0.115 \\ 0.707 & -0.577 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 7.07 & 1.7 \\ 0 & 0.346 \end{bmatrix}}_{R}$$

Problem 8: For each of the following, find a vector $\hat{\boldsymbol{x}}$ that minimizes $||A\hat{\boldsymbol{x}} - \boldsymbol{b}||$. Use the algorithm based on the QR factorization.

1. $A = \begin{bmatrix} 8 & 1 \\ 6 & 2 \\ 0 & 6 \end{bmatrix}$ and $\boldsymbol{b} = (10, 8, 6)$
2. $A = \begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix}$ and $\boldsymbol{b} = (10, 13)$

Linear regression

In this problem, you will find the "best" line through a given set of points, using QR factorization and solving a matrix equation where the matrix is upper triangular. (You can use the `solver` module if your `QR_solve(A,b)` procedure is not working.)

Module `read_data` defines a procedure `read_vectors(filename)` that takes a filename and reads a list of vectors from the named file.

The data we provide for this problem relates age to height for some young people in Kalam, an Egyptian village. Since children's heights vary a great deal, this dataset gives for each age from 18 to 29 the *average* height for the people of that age. The data is in the file `age-height.txt`

Problem 9: Use Python to find the values for parameters a and b defining the line $y = ax + b$ that best approximates the relationship between age (x) and height (y).

Least squares in machine learning

Ungraded problem: Try using the least-squares approach on the problem addressed in the machine learning lab. Compare the quality of the solution with that you obtained using gradient descent.