Problem1

In this problem, we want to explore a reflection transformation of X, which produces a mirror vector Z with respect to vector V. See Figure 1.

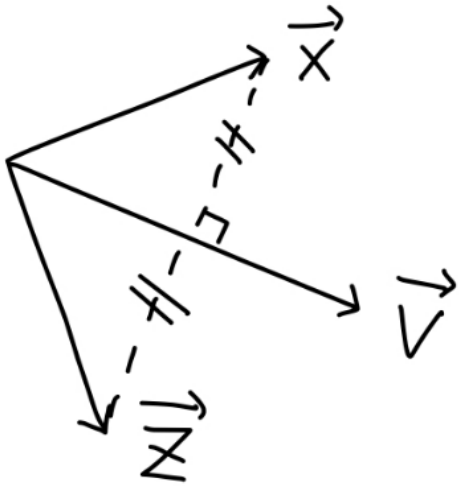


Figure 1

(a) Is a reflection transformation linear?

(b) If yes, find matrix M such that $Z = MX$ using concept of projection.

(c) If yes, find matrix M such that $Z = MX$ using concept of eigen values and vectors. (you can use the results of the above problems)

(d) For $V = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, compute M and its eigenvalues/eigenvectors (here, vector V is a vector with 45 degree angle with x-axis)

Problem 2

Let $R = R(\theta)$ be a rotation matrix with a rotational angle of θ in $\mathbb{R}^n$.

(a) Prove that

$$R^T R = I \text{ in } \mathbb{R}^n$$

- Hint: Euclidian distances of vectors are preserved after a rotational operation. (*i.e.*, $\|Rx\| = \|x\|$ for any $x \in \mathbb{R}^n$)

(b) Prove that

$$R^T(\theta) = R^{-1}(\theta) = R(-\theta) \text{ in } \mathbb{R}^n$$

(c) Show that column vectors in R are orthogonal in $\mathbb{R}^n$.

Problem3

Permutation matrices. A square matrix A is called a permutation matrix if it satisfies the following three properties:

- all elements of A are either zero or one
- each column of A contains exactly one element equal to one
- each row of A contains exactly one element equal to one.

The matrices

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

are examples of permutation matrices. A less formal definition is the following: a permutation matrix is the identity matrix with its rows reordered.

(a) Let A be an $n \times n$ permutation matrix. Give a simple description in words of the relation between a n-vector x and $f(x) = Ax$.

(b) Prove that columns of permutation matrix is orthogonal.

(c) We can also define a second linear function $g(x) = A^T x$ in terms of the same permutation matrix. What is the relation between g and f ?

- Hint: Use the result of (b)

(d) Describe meaning of columns and rows of permutation matrix. Then, explain the result of (c).

Problem 4

The regularized least-squares problem has the form

$$\min_{\theta} \|A\theta - y\|_2^2 + \lambda \|\theta\|_2^2$$

(a) Show that the solution is given by

$$\hat{\theta} = \left(A^T A + \lambda I_n\right)^{-1} A^T y$$

- Do not use the method of Lagrangian multipliers

(b) Write down a gradient descent algorithm for given optimization problem.

- Hint: Note that
  $$\|A\theta - y\|_2^2 = (A\theta - y)^T (A\theta - y)$$
  Then, you can differentiate above equation to compute the gradient. Likewise, you can compute the gradient of the regularizer.

(c) Based on the result of (b), describe the role of regularizer term.

- Hint: Gradient g is computed by $g = g_{projection} + g_{regularizer}$

(d) Describe results of (a) and (b) have the same meaning.

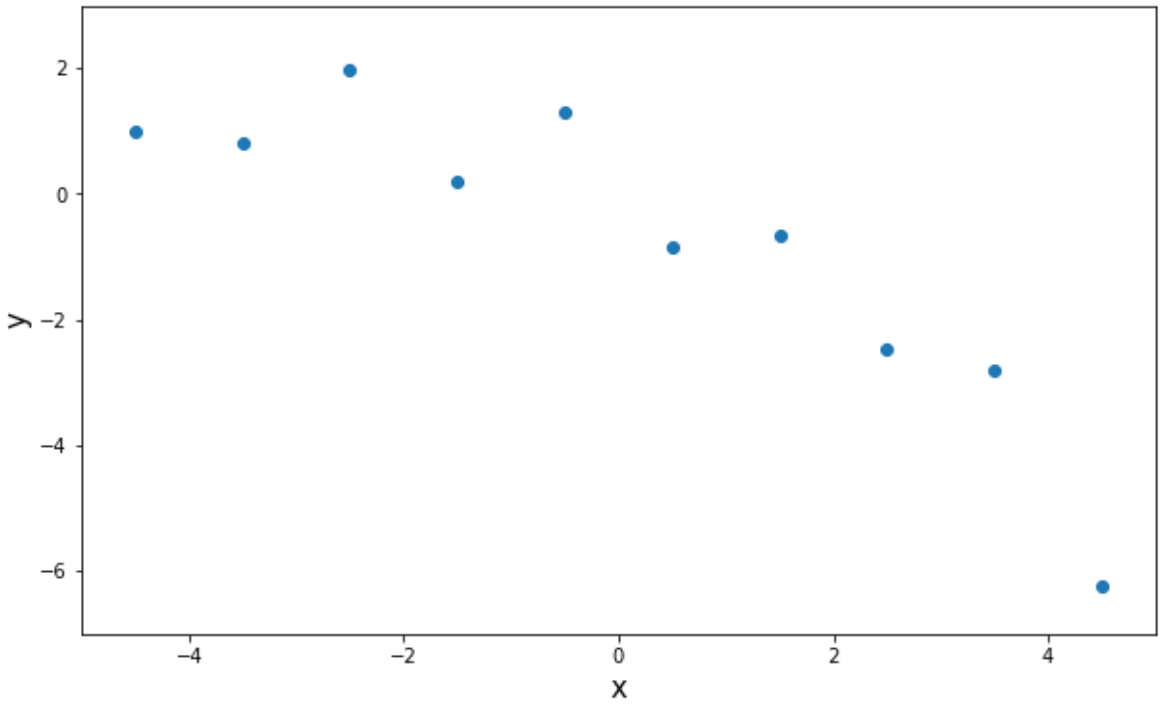
(e) Find and draw an approximated curve of the given data points in Python using your gradient descent algorithm.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-4.5, 4.5, 10)
y = np.array([0.9819, 0.7973, 1.9737, 0.1838, 1.3180, -0.8361, -0.6591, -2.4701, -2.8122, -6.2512])

plt.figure(figsize=(10, 6))
plt.plot(x, y, 'o')
plt.xlabel('x', fontsize=15)
plt.ylabel('y', fontsize=15)
plt.xlim(-5, 5)
plt.ylim(-7, 3)
plt.show()
```



Problem 5

a) Let's say x_i's are features. We say feature x_j is useless when

$$P(y|x_1, \dots x_{j-1}, x_j, x_{j+1}, \dots x_N) = P(y|x_1, \dots x_{j-1}, x_{j+1}, \dots x_N).$$

Describe the reason.

b) In our class, we learn that we can select meaningful feature using Lasso. Describe why Lasso select meaningful features using above equation.

- Hint: In our class, we assume $P(y|x) = sigmoid(x; \omega)$. Imagine the case where the ω is sparse.

c) Load 'data_input.pkl' and 'data_target.pkl' using followig command. Using these, build the regressor using Lasso.

In [2]:

```
from six.moves import cPickle
x = cPickle.load(open('./data/data_input.pkl', 'rb'))
y = cPickle.load(open('./data/data_target.pkl', 'rb'))
```

In [3]:

```
# Write your code here
```

d) After applying Lasso, we can select meaningful features. Then, we again apply Lasso to refine our features. Such techniques are called recursive feature elimination. Write the code that recursively apply Lasso to select features and plot the number of features selected at each iteration.

In [4]:

```
# Write your code here
```

Problem 6

(a) The figure shows loss functions of each classifier that you have learned in class.

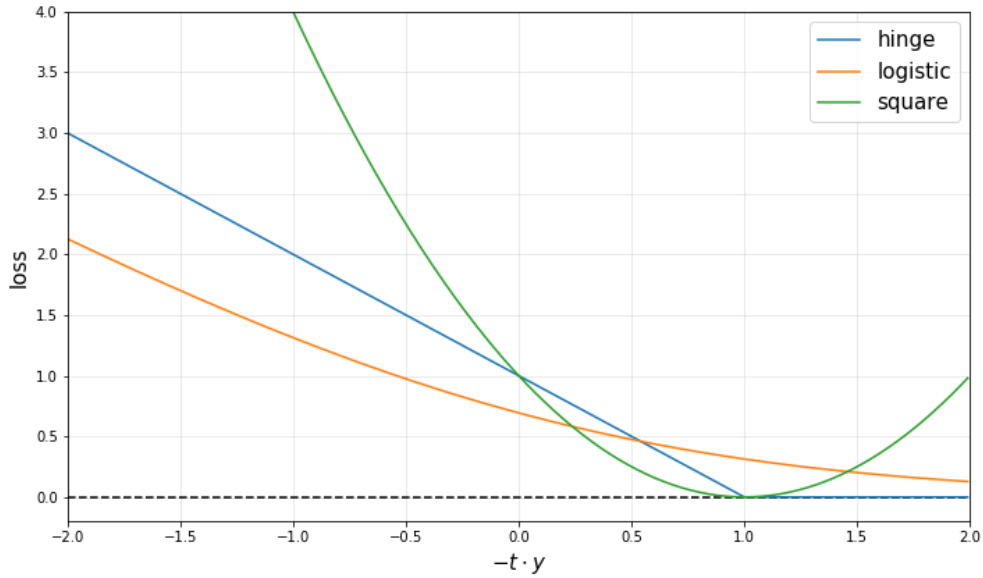


Figure 1

Here, hinge loss is defined by

$$hinge(y, t) = max(0, 1 - t \cdot y)$$

Observe that the hinge loss function does not impose any loss when the inner product between the target t and the prediction y is greater than 1. Actually, support vector machine can be coded by setting classifier's loss to the hinge loss function:

$$J(\omega) = \sum_{i=1}^{N} hinge(y^{(i)}, f(x^{(i)}; \omega))$$

where $x^{(i)}$ is the given input data and $y^{(i)}$ is the corresponding label. Describe the reason using the concept of maximum margin.

- Hint: Think what is the value of $y^{(i)} \cdot f(x^{(i)}; \omega)$ when the input data $x^{(i)}$ is inside the margin and outside the margin.

(b) Using the figure, describe why support vector machine is robust to outlier.

(c) Write the code that optimizing hinge loss function to build a classifier. Then plot the decision boundary of your classifier.

In [5]:

```
from six.moves import cPickle
x = cPickle.load(open('./data/data_input2.pkl', 'rb'))
y = cPickle.load(open('./data/data_target2.pkl', 'rb'))
```

In [6]:

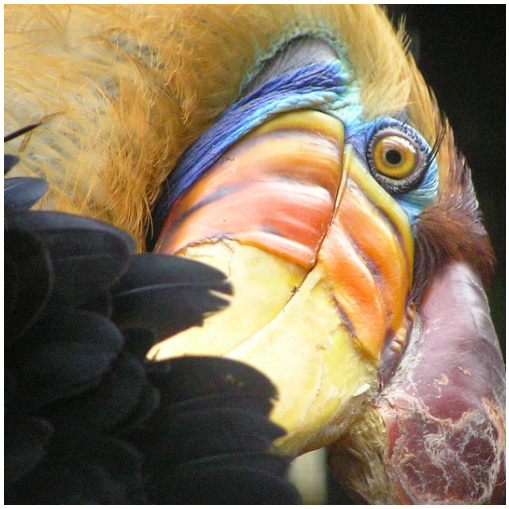
```
# Write you code here
```

Problem 7

You will use K-means to compress an image by reducing the number of colors it contains.

(a) Image Representation

The data for this exercise contains a 128-pixel by 128-pixel TIFF image named "bird.tiff." It looks like the picture in Figure 1.



In a straightforward 24-bit color representation of this image, each pixel is represented as three 8-bit numbers (ranging from 0 to 255) that specify red, green and blue (RGB) intensity values. Our bird photo contains thousands of colors, but we'd like to reduce that number to 16. By making this reduction, it would be possible to represent the photo in a more efficient way by storing only the RGB values of the 16 colors present in the image.

In this problem, you will use K-means to reduce the color count to $k = 16$. That is, you will compute 16 colors as the cluster centroids and replace each pixel in the image with its nearest cluster centroid color.

(b) K-means in Python

You can load a bird image using the following code.

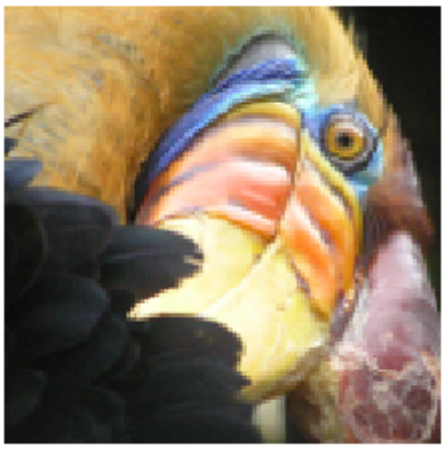
In [7]:

```
from six.moves import cPickle
import matplotlib.pyplot as plt

A = cPickle.load(open('./data/bird.pkl', 'rb'))

plt.figure(figsize=(4, 4))
plt.imshow(A.astype('uint8'))
plt.axis('off')
plt.show()

print('Matrix shape: {}'.format(A.shape))
```



```
Matrix shape: (128, 128, 3)
```

This creates a three-dimensional matrix A whose first two indices identify a pixel position and whose last index represents red, green, or blue. For example, $A(50, 33, 3)$ gives you the blue intensity of the pixel at position $y = 50, x = 33$. (The y-position is given first, but this does not matter so much in our example because the x and y dimensions have the same size).

Your task is to compute 16 cluster centroids from this image, with each centroid being a vector of length three that holds a set of RGB values. Here is the K-means algorithm as it applies to this problem:

(c) K-means algorithm

1. For initialization, sample 16 colors randomly from the original picture. There are your k means $\mu_1, \mu_2, \cdots, \mu_k$.

2. Go through each pixel in the small image and calculate its nearest mean.

$$c^{(i)} = \arg min_j \|x^{(i)} - \mu_j\|^2$$

3. Update the values of the means based on the pixels assigned to them.

$$\mu_j = \frac{\sum_i^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_i^m 1\{c^{(i)} = j\}}$$

4. Repeat steps 2 and 3 until convergence. This should take between 30 and 100 iterations. You can either run the loop for a preset maximum number of iterations, or you can decide to terminate the loop when the locations of the means are no longer changing by a significant amount.

Note: In Step 3, you should update a mean only if there are pixels assigned to it. Otherwise, you will see a divide-by-zero error. For example, it's possible that during initialization, two of the means will be initialized to the same color (*i.e.* black). Depending on your implementation, all of the pixels in the photo that are closest to that color may get assigned to one of the means, leaving the other mean with no assigned pixels.

When you have recalculated the image, you can display it. When you are finished, compare your image to the one in the solutions.

In [8]:

```
# Write down your code here
```