20180122_COGS118_Hw2

January 20, 2018

1 Matrix Calculus

1.1

$$
\begin{aligned}
f(x) &= \lambda(1 - x^2) \\
&= \lambda - \lambda x^2 \\
f'(x) &= -2\lambda x
\end{aligned}
$$

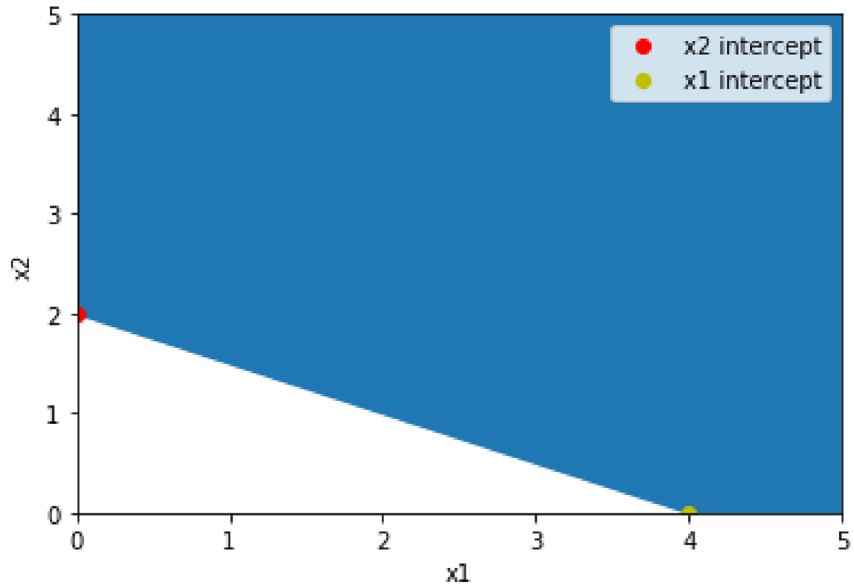
1.2

$$
\begin{aligned}
f(\mathbf{x}) &= \lambda(1 - \mathbf{x}^T A\mathbf{x}) \\
&= \lambda - \lambda\mathbf{x}^T A\mathbf{x} \\
\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial}{\partial \mathbf{x}}(\lambda) - \frac{\partial}{\partial \mathbf{x}}\left(\lambda\mathbf{x}^T A\mathbf{x}\right) \\
&= -\lambda\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^T A\mathbf{x}\right) \\
&= -2\lambda A\mathbf{x}
\end{aligned}
$$

2 Decision Boundary

2.1

```
In [44]: import matplotlib.pyplot as plt
         x = [0, 1 , 2, 3, 4, 5]
         y = [2, 1.5, 1, .5, 0, -.5]
         fig = plt.figure()
         fig.text(.5, -.01, "Plot of x1 + 2x2 - 4 > 0. Region shaded in blue represents where th
         plt.plot(x, y)
         plt.plot(0, 2, 'o', color='r', label='x2 intercept')
         plt.plot(4, 0, 'o', color='y', label='x1 intercept')
         plt.axis([0, 5, 0, 5])
         plt.ylabel('x2')
         plt.xlabel('x1')
         plt.fill_between(x, y, 10)
         plt.legend(loc = 'upper right')

         plt.show()
```



Plot of x1 + 2x2 - 4 > 0. Region shaded in blue represents where the classifier would predict 1.

2.2

When $x_1 = 0, x_2 = 1$ and when $x_1 = 1, x_2 = 0$. This gives three equations:

$$w_1 + b = 0$$
$$w_2 + b = 0$$
$$\sqrt{w_1^2 + w_2^2} = 1$$

Solving for w_1 and w_2 and plugging into the third equation we get:

$$\sqrt{(-b)^2 + (-b)^2} = 1$$
$$\sqrt{2b^2} = 1$$
$$b = \pm\frac{\sqrt{2}}{2}$$

Taking the negative sign for the above and plugging into the original equation we get:

$$\frac{\sqrt{2}}{2}x_1 + \frac{\sqrt{2}}{2}x_2 - \frac{\sqrt{2}}{2} \geq 0$$

We can verify this result by plugging in a coordinate that we know should predict 1, say (0, 2):

$$0 + \sqrt{2} - \frac{\sqrt{2}}{2} \geq 0$$
$$\frac{3\sqrt{2}}{2} \geq 0 \quad \checkmark$$

3 One-hot Encoding

Writing each sample as a row vector, we can represent S as a matrix like so,

$$S = \begin{pmatrix} 183 & 62 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 181 & 65 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 182 & 59 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 179 & 68 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 182 & 53 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The first column represents length in inches, the second column represents height in inches. Using one-hot encoding the next three columns represent the make of the car with a one in the third column representing a Toyota, a one in the fourth column representing BMW, and a one in the fifth column representing Ford. Also using one-hot encoding, the next 4 columns represent the color of the car with a 1 in the sixth column representing Blue, a one in the seventh column representing Silver, a one in the eighth column representing Red, and a one in the ninth column representing Black.

4 Conditional Probability

4.1

$$\begin{aligned} P(cancer+|test+) &= \frac{P(test+|cancer+)P(cancer+)}{P(test+|cancer+)P(cancer+) + P(test+|cancer-)P(cancer-)} \\ &= \frac{(.98)(.0006)}{(.98)(.0006) + (.06)(.9994)} \\ &= 0.00971066191 \\ &\approx .97\% \end{aligned}$$

4.2

$$\begin{aligned} P(cancer-|test-) &= \frac{P(test-|cancer-)P(cancer-)}{P(test-|cancer-)P(cancer-) + P(test-|cancer+)P(cancer+)} \\ &= \frac{(.94)(.9994)}{(.94)(.9994) + (.02)(.0006)} \\ &= 0.99998722654 \\ &\approx 99.99\% \end{aligned}$$

4.3

Precision:

$P(cancer+|test+) = .97\%$

Recall:

$P(test+|cancer+) = 98\%$

F-value:

$$
\begin{aligned}
\frac{2(Precision \times Recall)}{Precision + Recall} &= \frac{2(0.0097 \times 0.98)}{0.0097 + 0.98} \\
&= \frac{2(0.009506)}{0.9897} \\
&= \frac{0.019012}{0.9897} \\
&= 0.01920986157
\end{aligned}
$$

5 Binary Communication System

5.1

$$P(X = 2) = 0$$

5.2

$$P(Y = 0|X = 1) = P(Y = 0, X = 1)P(X = 1) = (.3)(.8) = .24$$

5.3

$$P(Y = 0) = (.3)(.8) + (.5)(.2) = .34$$

5.4

$$
\begin{aligned}
P(X = 1|Y = 0) &= \frac{P(Y = 0|X = 1)P(X = 1)}{P(Y = 0)} \\
&= \frac{.24}{.34} \\
&= 0.7058823529411764 \\
&\approx 0.7059
\end{aligned}
$$

Decision Stump

January 20, 2018

0.1 Decision Stump

In this problem, we will perform a binary classification task on the Iris dataset. This dataset has 150 data points, where each data point $\mathbf{x} \in \mathbb{R}^4$ has 4 features and its corresponding label $y \in \{0, 1\}$.

(In fact, the original Iris dataset has 3 classes: 0 for Setosa, 1 for Versicolor and 2 for Virginica. Here for binary classification task, we combine Setosa and Versicolor together as $y = 0$ and label Virginca as $y = 1$)

To classify these 2 labels above, we decide to utilize a decision stump. The decision stump works as follows (for simplicity, we restrict our attention to uni-directional decision stumps):

- Given the j-th feature $\mathbf{x}(j)$ and a threshold Th, for each data point with index i, the classification function is defined by $y = f(\mathbf{x}, j, Th)$ as:

$$f(\mathbf{x}, j, Th) = \begin{cases} 1 & if\ \mathbf{x}(j) \geq Th \\ 0 & otherwise \end{cases}$$

Based on the decision stump above, we wish to use an algorithm to find the **best feature** and **best threshold** on training set to create a "best" decision stump, in a sense that such decision stump can achieve the **highest accuracy on training set**:

- Loop over j-th feature $\mathbf{x}(j)$ (j = 0, 1, 2, 3).
- Loop over all possible threshold Th between the minimum and maximum of $\mathbf{x}(j)$.
  1. For each data point $\mathbf{x}(j)$ with data point index i ($i = 0, \ldots, 99$) (the first 100 points for training set), predict:

$$y \Rightarrow 0 \text{ if } \mathbf{x}(j) \leq Th, \qquad y \Rightarrow 1 \text{ if } \mathbf{x}(j) > Th$$

  2. Calculate the accuracy over the training set.
- Output feature index j and threshold Th, which achieves the best accuracy.

Please fill the function **calc_acc(Xj, Y, thres)** in *1.3 Find the best feature and best threshold*.

The first histogram printed in 1.3 should be like:

If you use the PDF of this notebook, just fill the function, run the whole notebook and save all contents here. Otherwise, if you want to create a separate document, please include:

- All 4 histograms in last part of the code.
- The best feature, best threshold, training and test accuracy in last part of the code.

```
In [14]: %config InlineBackend.figure_format = 'retina'
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn import datasets
```

0.1.1 Load the Iris dataset

```
In [15]: # Iris dataset.
         iris = datasets.load_iris()      # Load Iris dataset.

         X = iris.data                    # The shape of X is (150, 4), which means
                                          # there are 150 data points, each data point
                                          # has 4 features.

         # Here for convenience, we divide the 3 kinds of flowers into 2 groups:
         #     Y = 0 (or False):  Setosa (original value 0) / Versicolor (original value 1)
         #     Y = 1 (or True):   Virginica (original value 2)

         # Thus we use (iris.target > 1.5) to divide the targets into 2 groups.
         # This line of code will assign:
         #    Y[i] = True  (which is equivalent to 1) if iris.target[k]  > 1.5 (Virginica)
         #    Y[i] = False (which is equivalent to 0) if iris.target[k] <= 1.5 (Setosa / Versico

         Y = (iris.target > 1.5).reshape(-1,1) # The shape of Y is (150, 1), which means
                                          # there are 150 data points, each data point
                                          # has 1 target value.

         X_and_Y = np.hstack((X, Y))      # Stack them together for shuffling.
         np.random.seed(1)                # Set the random seed.
         np.random.shuffle(X_and_Y)       # Shuffle the data points in X_and_Y array

         print(X.shape)
         print(Y.shape)
         print(X_and_Y.shape)
         print(X_and_Y[0])                # The result should be always: [ 5.8  4.   1.2  0.2  0.

(150, 4)
(150, 1)
(150, 5)
[ 5.8  4.   1.2  0.2  0. ]
```

```
In [16]: # Divide the data points into training set and test set.
         X_shuffled = X_and_Y[:,:4]
         Y_shuffled = X_and_Y[:,4]
```

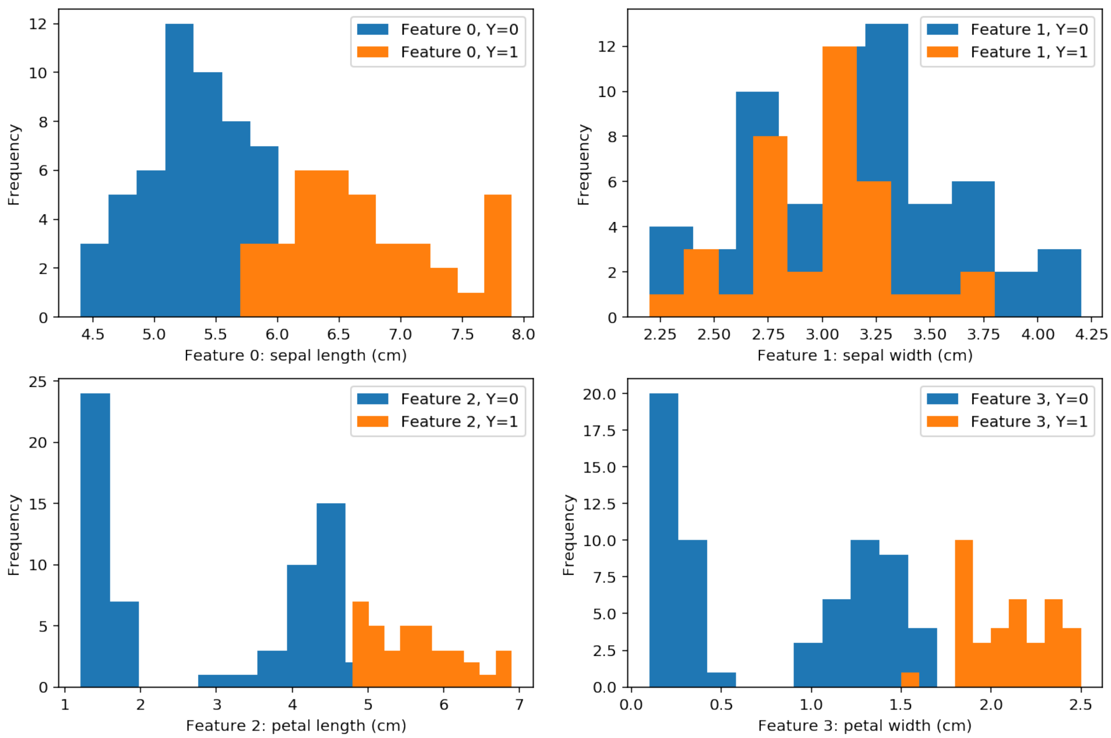
```
X_train = X_shuffled[:100] # Shape: (100,4)
Y_train = Y_shuffled[:100] # Shape: (100,)
X_test = X_shuffled[100:]  # Shape: (50,4)
Y_test = Y_shuffled[100:]  # Shape: (50,)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(100, 4)
(100,)
(50, 4)
(50,)
```

0.1.2 Draw the histograms of each feature

In [17]:
```
# Show the histograms of each feature.
plt.figure(figsize=(12,8))
for j in range(4):
    Xj_train = X_train[:,j]
    Xj_when_Y0_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_train[i] == 0]
    Xj_when_Y1_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_train[i] == 1]

    plt.subplot(2, 2, j+1)
    plt.hist(Xj_when_Y0_train, label='Feature {}, Y=0'.format(j))
    plt.hist(Xj_when_Y1_train, label='Feature {}, Y=1'.format(j))
    plt.xlabel('Feature {}: {}'.format(j, iris.feature_names[j]))
    plt.ylabel('Frequency')
    plt.legend()
plt.show()
```

```
<matplotlib.figure.Figure at 0x7f3ca83bd898>
```



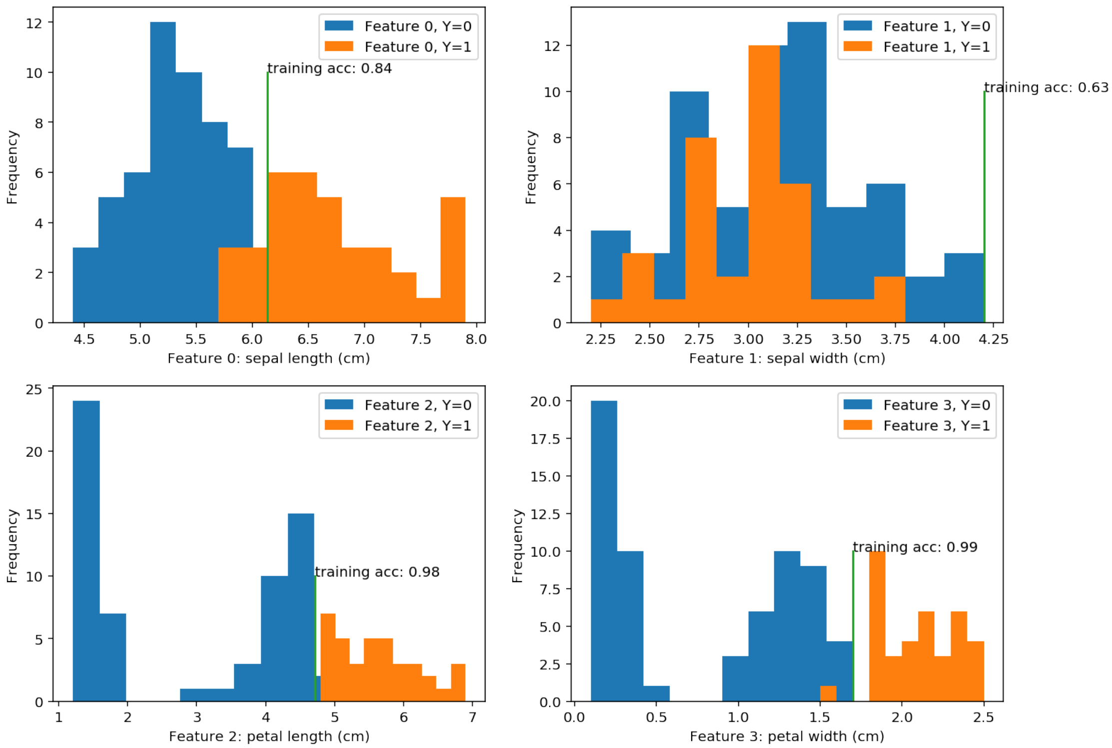
0.1.3 Find the best feature and best threshold

In [18]:
```
# Calculate the accuracy of prediction given feature, target and threshold.
def calc_acc(Xj, Y, thres):
    """
    Calculate the accuracy given feature, target and threshold.
        Xj:     j-th feature. This array only contains 1 feature for all data points,
                so the shape should be (count of data points,)
        Y:      Target array. Shape: (count of data points,)
        thres:  Threshold.
    Return the accuracy of prediction.
    """
    # Step 1. Count the number of correct predictions and incorrect predictions.
    #         Here, for simplicity, we assume:
    #              If feature <= threshold, we predict it as Y = 0.
    #              If feature  > threshold, we predict it as Y = 1.
    n_correct = 0
    n_incorrect = 0

    for i, x in enumerate(Xj):
        if x <= thres:
            if Y[i] == 0:
                n_correct += 1
```

```
            else:
                n_incorrect += 1
        else:
            if Y[i] == 1:
                n_correct += 1
            else:
                n_incorrect += 1

    # Step 2. Calculate the accuracy.
    acc = 1.0 * n_correct / (n_correct + n_incorrect)

    return acc
```

In [19]:
```
# Show the histograms of each feature.
plt.figure(figsize=(12,9))

all_max_acc = 0.0  # Max training accuracy among all features.
all_thres   = None # Threshold when reach the max training accuracy.
all_feature = None # Index of feature when reach the max training accuracy.

# Loop over 4 features. j: index of current feature.
for j in range(4):
    # Get data.
    Xj_train = X_train[:,j] # Array of feature j.
    Xj_when_Y0_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_train[i] == 0] #
    Xj_when_Y1_train = [Xj_train[i] for i in range(len(Xj_train)) if Y_train[i] == 1] #

    current_max_acc = 0.0    # Max training accuracy in current feature.
    current_thres   = None   # Threshold when reach the max accuracy in current feature

    # Loop over all possible values for threshold. Here we consider 100 numbers between
    for thres in np.linspace(Xj_train.min(), Xj_train.max(), 100):
        # Calculate the accuracy on training data given feature, target and threshold.
        acc = calc_acc(Xj_train, Y_train, thres)
        # Update the current max accuracy if possible.
        if acc > current_max_acc:
            current_max_acc = acc
            current_thres = thres

    # Update the max training accuracy among all features if possible.
    if current_max_acc > all_max_acc:
        all_max_acc = current_max_acc
        all_thres   = current_thres
        all_feature = j

    # Plot the histograms and the best decision stump in current feature.
    plt.subplot(2, 2, j+1)
    plt.hist(Xj_when_Y0_train, label='Feature {}, Y=0'.format(j))
```

```
        plt.hist(Xj_when_Y1_train, label='Feature {}, Y=1'.format(j))
        plt.plot([current_thres, current_thres], [0, 10])
        plt.text(current_thres, 10, 'training acc: {}'.format(current_max_acc))
        plt.xlabel('Feature {}: {}'.format(j, iris.feature_names[j]))
        plt.ylabel('Frequency')
        plt.legend()
    plt.show()
```



In [20]:
```
# Use the best feature and best threshold on test set.

Xj_test = X_test[:, all_feature] # Array of best feature.
test_acc = calc_acc(Xj_test, Y_test, all_thres)
print('Best feature: {}'.format(all_feature))
print('Best threshold: {:.2f}'.format(all_thres))
print('Training accuracy of best feature: {:.2f}'.format(all_max_acc))
print('Test accuracy of best feature: {:.2f}'.format(test_acc))
```

```
Best feature: 3
Best threshold: 1.70
Training accuracy of best feature: 0.99
Test accuracy of best feature: 0.90
```