Last Edited: 2024-01-25

Changes:

- (01-25) Added code to remove 7 datapoints that cause log(0) to the `fashion_mnist()` function
- (01-25) Fixed some typos

- **Deadline**: Feb 5, at 23:59PM.
- **Submission**: You need to submit your solutions through Crowdmark, including all your derivations, plots, and your code. You can produce the files however you like (e.g. LATEX, Microsoft Word, etc), as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code.
- **Collaboration policy**: After attempting the problems on an individual basis, you may discuss and work together on the assignment with up to two classmates. However, **you must write your own code and write up your own solutions individually and explicitly name any collaborators** at the top of the homework.

Q1 - Decision Theory

One successful use of probabilistic models is for building spam filters, which take in an email and take different actions depending on the likelihood that it's spam.

Imagine you are running an email service. You have a well-calibrated spam classifier that tells you the probability that a particular email is spam: $p(spam|email)$. You have four options for what to do with each email: You can list it as important email, show it to the user, put it in the spam folder, or delete it entirely.

Depending on whether or not the email really is spam, the user will suffer a different amount of wasted time for the different actions we can take, L(action, spam):

Action	Spam	Not spam
Important	15	0
Show	5	1
Folder	1	40
Delete	0	150

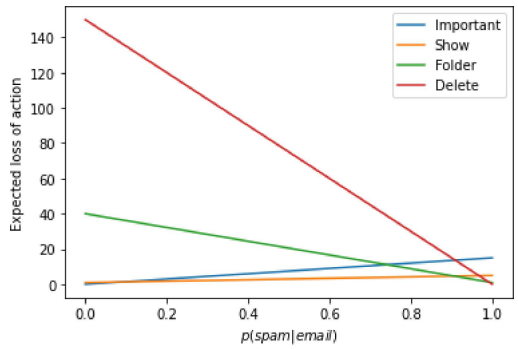
Double-click (or enter) to edit

Q1.1

[3pts] Plot the expected wasted user time for each of the three possible actions, as a function of the probability of spam: $p(spam|email)$.

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


losses = [[15, 0],[5, 1], [1, 40],[0, 150]]
actions_names = ['Important','Show', 'Folder', 'Delete']
num_actions = len(losses)
def expected_loss_of_action(prob_spam, action):
    #TODO: Return expected loss over a Bernoulli random variable
    # with mean prob_spam.
    # Losses are given by the table above.

    return prob_spam * losses[action][0] + (1 - prob_spam) * losses[action][1]

prob_range = np.linspace(0., 1., num=600)

# Make plot
for action in range(num_actions):
    plt.plot(prob_range, expected_loss_of_action(prob_range, action), label=actions_names[action])

plt.xlabel('$p(spam|email)$')
plt.ylabel('Expected loss of action')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f116f76e040>
```



Q1.2

[2pts] Write a function that computes the optimal action given the probability of spam.

```
def optimal_action(prob_spam):
    #TODO: return best action given the probability of spam.
    #Hint: np.argmin might be helpful.
    min_loss = float('inf')
    min_loss_action = None
    for action in range(num_actions): #Loop through all actions
        loss = expected_loss_of_action(prob_spam, action) #Find expected loss
        if loss < min_loss:
            min_loss = loss #Save best loss and action
            min_loss_action = action
    return min_loss_action
```

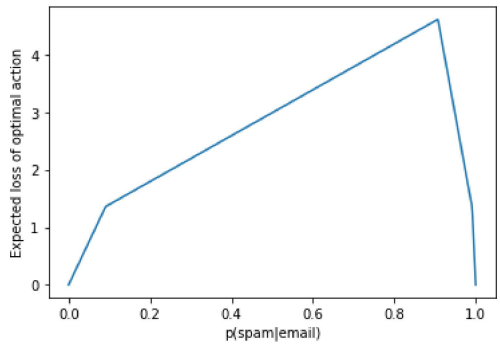
Q1.3

[4pts] Plot the expected loss of the optimal action as a function of the probability of spam.

Color the line according to the optimal action for that probability of spam.

```
prob_range = np.linspace(0., 1., num=600)
optimal_losses = []
optimal_actions = []
for p in prob_range: #Loop through probability range
    # TODO: Compute the optimal action and its expected loss for
    # probability of spam given by p.
    opt_action = optimal_action(p) #Find optimal action
    opt_loss = expected_loss_of_action(p, opt_action) #Find expected loss
    optimal_actions.append(opt_action)
    optimal_losses.append(opt_loss)

plt.xlabel('p(spam|email)')
plt.ylabel('Expected loss of optimal action')
plt.plot(prob_range, optimal_losses)
```

```
[<matplotlib.lines.Line2D at 0x7f116f865a90>]
```



Q1.4

[4pts] For exactly which range of the probabilities of an email being spam should we delete an email?

Find the exact answer by hand using algebra.

Range where deleting an email is better than listing as important email:
15p > 150(1-p)
15p > 150-150p
165p > 150
p > 150/165

Range where deleting an email is better than show the email to the user:
5p+(1-p) > 150(1-p)
4p+1 > 150-150p
154p > 149
p > 149/154

Range where deleting an email is better than putting the email in spam folder:
p+40(1-p) > 150(1-p)
p+40-40p > 150-150p
111p > 110
p > 110/111

Putting the email in spam folder is the limiting factor so range of probabilities of an email being spam should be deleted is $110/111 < p \leq 1$.

Q2 - Naïve Bayes, A Generative Model