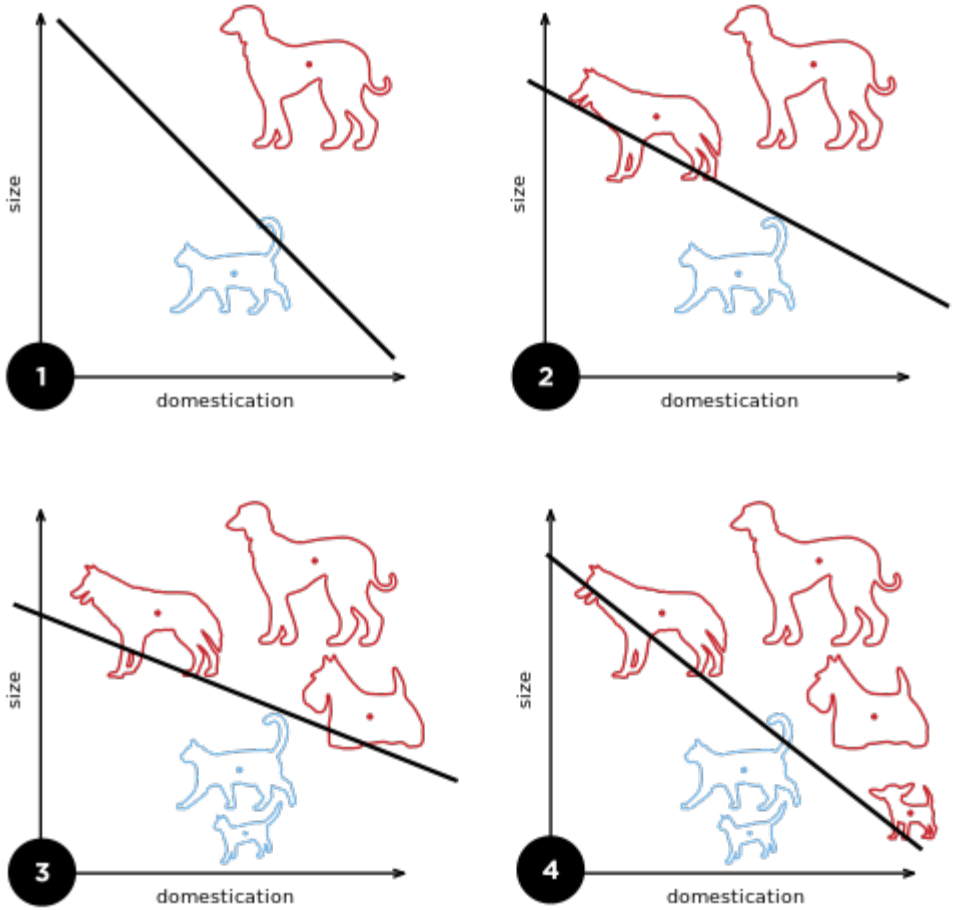
Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:
```
NAME = ""
COLLABORATORS = ""
```


size
1
domestication
size
2
domestication
size
3
domestication
size
4
domestication

Optimization for classification

$$y = mx + c$$

$$e(y_i, x_i; m, c) = \begin{cases} 0 & \text{if } mx_i + c = l_i \\ |mx_i + c| & \text{if } mx_i + c \neq l_i \end{cases}$$

$$e(y_i, x_i; m, c) = \begin{cases} 0 & \text{if } mx_i + c = l_i \\ |mx_i + c| & \text{if } mx_i + c \neq l_i \end{cases}$$

$$\mathbf{m} = \begin{bmatrix} m \\ c \end{bmatrix}$$

$$e(y_i, x_i; \mathbf{m}) = \begin{cases} 0 & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} = l_i \\ |\begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m}| & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} \neq l_i \end{cases}$$

$$\nabla_{\mathbf{m}} e(y_i, x_i; \mathbf{m}) = \begin{cases} 0 & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} = l_i \\ |\begin{bmatrix} x_i & 1 \end{bmatrix}| & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} \neq l_i \end{cases}$$

If $l_i \in \{-1, 1\}$, then we can write

$$e(y_i, x_i; \mathbf{m}) = \max\{0, -l_i \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m}\}$$

$$\nabla_{\mathbf{m}} e(y_i, x_i; \mathbf{m}) = \max\{0, -l_i \begin{bmatrix} x_i & 1 \end{bmatrix}\}$$

$$\mu_x(I) = \sum_{x=1}^{W} \frac{xI(x,y)}{\sum_{x=1}^{W} I(x,y)}$$

$$\sigma_x^2(I) = \sum_{x=1}^{W} \frac{(x-\mu_x)^2 I(x,y)}{\sum_{x=1}^{W} I(x,y)}$$

In [ ]:
```
def error(X, Y, bfm):
    # YOUR CODE HERE
    raise NotImplementedError()

def grad_error(Xw, Yw, bfm):
    # YOUR CODE HERE
    raise NotImplementedError()

def train(X, Y, lr = 0.1):
    # YOUR CODE HERE
    raise NotImplementedError()

OPTIMAL_BFM, list_of_bfms, list_of_errors = train(X, Y)
fig, ax = plt.subplots()
ax.plot(list_of_errors)
ax.set_xlabel('t')
ax.set_ylabel('loss')
plt.show()
```

In [ ]:
```
positive_label = 1
negative_label = 0
TP = np.sum((zero_one_test_labels == positive_label) & (zero_one_predic
TP
```

In [ ]:
```
TN =  np.sum((zero_one_test_labels == negative_label) & (zero_one_predi
TN
```

In [ ]:
```
FP = np.sum((zero_one_test_labels != positive_label) & (zero_one_predic
FP
```

In [ ]:
```
FN = np.sum((zero_one_test_labels != negative_label) & (zero_one_predic
FN
```

In [ ]:
```
# Confusion matrix
fig, ax = plt.subplots()
ax.imshow([[TN, FN],
           [FP, TP]])
ax.set_xlabel('predicted')
ax.set_ylabel('true')
ax.axis('off')
```

Next

2. Show visualization of 1D optimization and loss functions.
3. Build to visualizations in the UDL book. Connect to KD tree and nearest neighbor classification.
4. Show the tensflow js visualization.

References

1. http://playground.tensorflow.org
2. https://knowyourdata-tfds.withgoogle.com/#tab=STATS&dataset=tf_flowers
3. "Flowers", The TensorFlow Team. Jan 2019. Online http://download.tensorflow.org/example_images/flower_photos.tgz