Midterm Project: Regularized Logistic Regression with Real Dataset

An extension of Logistic Regression with a real dataset

Goals

```
1. Upload and transform real valued dataset
2. Scale the dataset using z-score normalization
3. Implement regularization (extending compute_cose and compute_gradient f
unctions)
4. Plot the learning curve (cost vs iterations)
```

Packages

First, we must import the required packages

In [1]:
```
import copy, math
import numpy as np
import matplotlib.pyplot as plt
import time
```

The Dataset

This dataset is a dataset of diagnostic breast cancer data which includes 30 real values input features. These features encompass many medical attributes about each patient's tumor. Specifically, the 30 real valued features are computed from 10 attributes about each cell in the tumor. For each of the 10 attributes, the mean, standard error, and mean of the three largest values are recorded, resulting in 30 input features for our regression model. In addition to these features, each data point has an id number and a result, whether the tumor was malignant or benign.

Here we load this dataset:

- `X_train` contains the 30 real valued input features
- `y_train` is the diagnostic decision
    - `y_train = 1` if the patient's tumor was malignant
    - `y_train = 0` if the patient's tumor was benign
- Both `X_train` and `y_train` are numpy arrays.

In [2]:
```
def load_data(filename):
    data = np.loadtxt(filename, dtype=str, delimiter=',')

    X = data[:,2:].astype(np.float)
    y = [1 if item == 'M' else 0 for item in data[:,1]]

    return X, y
```

In [3]:
```
# load dataset

X_train, y_train = load_data("./data/wdbc.data")
```

Notation

Here is a summary of some of the notation you will encounter, updated for multiple features.

General Notation	Description	Python (if applicable)
a	scalar, non bold	
$\mathbf{a}$	vector, bold	
$\mathbf{A}$	matrix, bold capital	
Regression		
$\mathbf{X}$	training example maxtrix	`X_train`
$\mathbf{y}$	training example targets	`y_train`
$\mathbf{x}^{(i)}$, $y^{(i)}$	i_{th} Training Example	`X[i]`, `y[i]`
m	number of training examples	`m`
n	number of features in each example	`n`
$\mathbf{w}$	parameter: weight,	`w`
b	parameter: bias	`b`
$f_{\mathbf{w},b}(\mathbf{x}^{(i)})$	The result of the model evaluation at $\mathbf{x^{(i)}}$ parameterized by $\mathbf{w},b$: $f_{\mathbf{w},b}(\mathbf{x}^{(i)}) = \mathbf{w} \cdot \mathbf{x}^{(i)}+b$	`f_wb`

Normalization

Here we will use the z score normalization technique to normalize the dataset.

After z-score normalization, all features will have a mean of 0 and a standard deviation of 1.

To implement z-score normalization, adjust your input values as shown in this formula:

$$x^{(i)}_j = \dfrac{x^{(i)}_j - \mu_j}{\sigma_j} \tag{4}$$

where j selects a feature or a column in the $\mathbf{X}$ matrix. μ_j is the mean of all the values for feature (j) and σ_j is the standard deviation of feature (j).

$$\mu_j = \frac{1}{m} \sum_{i=0}^{m-1} x^{(i)}_j \tag{5}$$

$$\sigma^2_j = \frac{1}{m} \sum_{i=0}^{m-1} (x^{(i)}_j - \mu_j)^2 \tag{6}$$

In [4]:
```
def zscore_normalize_features(X):
    """
    computes  X, zcore normalized by column

    Args:
      X (ndarray (m,n))     : input data, m examples, n features

    Returns:
      X_norm (ndarray (m,n)): input normalized by column
      mu (ndarray (n,))     : mean of each feature
      sigma (ndarray (n,))  : standard deviation of each feature
    """
    # find the mean of each column/feature
    mu     = np.mean(X, axis=0)                 # mu will have shape (n,)
    # find the standard deviation of each column/feature
    sigma  = np.std(X, axis=0)                  # sigma will have shape (n,)
    # element-wise, subtract mu for that column from each example, divide by std for that column
    X_norm = (X - mu) / sigma

    return X_norm
```

In [5]:
```
# normalize the original features
X_norm = zscore_normalize_features(X_train)
```

Regularization

Here we will extend the compute cost and compute gradient functions to utilize regularization techniques to avoid overfitting.

Cost function and regression function for Regularized Logistic Regression:

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}w_j^2$$

$$\text{repeat}\ \{$$

$$w_j = w_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)-y^{(i)}\right)x_j^i\right] + \frac{\lambda}{m}w_j\right]$$

$$b = b - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)-y^{(i)}\right)\right]$$

$$\}$$

$$\text{Where } f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

In [6]:
```
def sigmoid(z):
    """
    Compute the sigmoid of z

    Args:
        z (ndarray): A scalar, numpy array of any size.

    Returns:
        g (ndarray): sigmoid(z), with the same shape as z

    """
    g = (1/(1+np.exp(-z)))
    return g

def compute_cost(X, y, w, b, lambda_= 1):
    """
    Computes the cost over all examples
    Args:
      X : (ndarray Shape (m,n)) data, m examples by n features
      y : (array_like Shape (m,)) target value
      w : (array_like Shape (n,)) Values of parameters of the model
      b : scalar Values of bias parameter of the model
      lambda_: scalar value for regularization
    Returns:
      total_cost: (scalar)         cost
    """

    m, n = X.shape

    total_cost = ((-1/m)*sum_losses(X, y, w, b, m))+((lambda_/(2*m))
*sum_of_squared_features(w, n))


    return total_cost

def sum_of_squared_features(w, n):
    return sum([w[j]**2 for j in range(n)])


def sum_losses(X, y, w, b, m):
    return sum([loss(sigmoid(np.dot(w, X[i])+b), y[i]) for i in rang
e(m)])


def loss(fwbx, y):
    return (y*np.log(fwbx)) + (1-y)*np.log(1-fwbx)
```

Gradient Descent

Here we calculate the gradient for logistic regression with regularization and use it to calculate the gradient descent to learn theta.

In [7]:
```
def compute_gradient(X, y, w, b, lambda_=None):
    """
    Computes the gradient for logistic regression

    Args:
      X : (ndarray Shape (m,n)) variable such as house size
      y : (array_like Shape (m,1)) actual value
      w : (array_like Shape (n,1)) values of parameters of the model
      b : (scalar)                 value of parameter of the model
      lambda_: scalar value for regularization
    Returns
      dj_dw: (array_like Shape (n,1)) The gradient of the cost w.r.
t. the parameters w.
      dj_db: (scalar)                The gradient of the cost w.r.t.
the parameter b.
    """
    m, n = X.shape
    dj_dw = np.zeros(w.shape)
    dj_db = 0.

    dj_db = (1/m)*sum([sigmoid(np.dot(X[i], w)+b)-y[i] for i in rang
e(m)])

    dj_dw = (1/m)*sum([(sigmoid(np.dot(X[i], w)+b)-y[i])*X[i] for i
in range(m)])+ ((lambda_/m)*w)


    return dj_db, dj_dw

def gradient_descent(X, y, w_in, b_in, cost_function, gradient_funct
ion, alpha, num_iters, lambda_):
    """
    Performs batch gradient descent to learn theta. Updates theta by
taking
    num_iters gradient steps with learning rate alpha

    Args:
      X :    (array_like Shape (m, n)
      y :    (array_like Shape (m,))
      w_in : (array_like Shape (n,))  Initial values of parameters o
f the model
      b_in : (scalar)                 Initial value of parameter of
the model
      cost_function:                  function to compute cost
      alpha : (float)                 Learning rate
      num_iters : (int)               number of iterations to run gr
adient descent
      lambda_ (scalar, float)         regularization constant

    Returns:
      w : (array_like Shape (n,)) Updated values of parameters of th
e model after
          running gradient descent
      b : (scalar)                Updated value of parameter of the
model after
```

```
          running gradient descent
    """

    # number of training examples
    m = len(X)

    # An array to store cost J and w's at each iteration primarily f
or graphing later
    J_history = []
    w_history = []

    for i in range(num_iters):

        # Calculate the gradient and update the parameters
        dj_db, dj_dw = gradient_function(X, y, w_in, b_in, lambda_)

        # Update Parameters using w, b, alpha and gradient
        w_in = w_in - alpha*dj_dw
        b_in = b_in - alpha*dj_db

        # Save cost J at each iteration
        if i<100000:      # prevent resource exhaustion
            cost =  cost_function(X, y, w_in, b_in, lambda_)
            J_history.append(cost)

        # Print cost every at intervals 10 times or as many iteratio
ns if < 10
        if i% math.ceil(num_iters/10) == 0 or i == (num_iters-1):
            w_history.append(w_in)
            print(f"Iteration {i:4}: Cost {float(J_history[-1]):8.2
f}   ")

    return w_in, b_in, J_history, w_history #return w and J,w histor
y for graphing
```

Training

Here we run gradient descent for our model with some parameters we can experiment with to get the best results. A larger alpha means that the weights and biases are modified more at every step. A larger lambda means more regulization is done.

In [8]:
```
# initialize parameters
initial_w = np.zeros((X_norm.shape[1]))
initial_b = 0.
# some gradient descent settings
iterations = 10000
alpha = 1.0e-3
lambda_ = 5
# run gradient descent
w_final, b_final, J_hist, W_hist = gradient_descent(X_norm, y_train, initial_w, initial_b,
                                                    compute_cost, compute_gradient,
                                                    alpha, iterations, lambda_)
print(f"b,w found by gradient descent: {b_final:0.2f},{w_final} ")
m,_ = X_train.shape
for i in range(m):
    print(f"prediction: {np.dot(X_norm[i], w_final) + b_final:0.2f}, target value: {y_train[i]}")
```

```
Iteration    0: Cost     0.69
Iteration 1000: Cost     0.25
Iteration 2000: Cost     0.19
Iteration 3000: Cost     0.16
Iteration 4000: Cost     0.15
Iteration 5000: Cost     0.14
Iteration 6000: Cost     0.13
Iteration 7000: Cost     0.13
Iteration 8000: Cost     0.12
Iteration 9000: Cost     0.12
Iteration 9999: Cost     0.12
b,w found by gradient descent: -0.33,[ 0.36871781  0.32313821  0.365
11082  0.36364845  0.13722045  0.13426724
  0.28130855  0.37135542  0.10100307 -0.1566099   0.32108198  0.0023
1499
  0.27656912  0.29478144 -0.00445448 -0.08629911 -0.07214296  0.0609
4345
 -0.06199763 -0.16763157  0.43350334  0.39224404  0.41734921  0.4081
4405
  0.29043102  0.2011423   0.27759468  0.3924805   0.26841951  0.0987
233 ]
prediction: 9.79, target value: 1
prediction: 4.52, target value: 1
prediction: 7.53, target value: 1
prediction: 4.59, target value: 1
prediction: 4.65, target value: 1
prediction: 1.32, target value: 1
prediction: 4.44, target value: 1
prediction: 1.76, target value: 1
prediction: 3.11, target value: 1
prediction: 4.93, target value: 1
prediction: 0.66, target value: 1
prediction: 3.28, target value: 1
prediction: 6.45, target value: 1
prediction: 0.33, target value: 1
prediction: 3.00, target value: 1
prediction: 4.73, target value: 1
prediction: 1.59, target value: 1
prediction: 5.58, target value: 1
prediction: 8.68, target value: 1
prediction: -1.78, target value: 0
prediction: -2.70, target value: 0
prediction: -6.22, target value: 0
prediction: 3.49, target value: 1
prediction: 8.54, target value: 1
prediction: 8.00, target value: 1
prediction: 7.39, target value: 1
prediction: 4.21, target value: 1
prediction: 4.19, target value: 1
prediction: 5.87, target value: 1
prediction: 1.90, target value: 1
prediction: 8.36, target value: 1
prediction: 2.26, target value: 1
prediction: 6.19, target value: 1
prediction: 7.38, target value: 1
prediction: 3.80, target value: 1
```

```
prediction: 4.46, target value: 1
prediction: 1.43, target value: 1
prediction: -4.79, target value: 0
prediction: -0.47, target value: 1
prediction: 0.78, target value: 1
prediction: -1.58, target value: 1
prediction: -0.13, target value: 1
prediction: 9.39, target value: 1
prediction: 1.22, target value: 1
prediction: 0.55, target value: 1
prediction: 6.66, target value: 1
prediction: -6.77, target value: 0
prediction: 1.81, target value: 1
prediction: -2.99, target value: 0
prediction: -1.24, target value: 0
prediction: -4.14, target value: 0
prediction: -3.79, target value: 0
prediction: -4.49, target value: 0
prediction: 4.09, target value: 1
prediction: 0.68, target value: 1
prediction: -3.71, target value: 0
prediction: 7.38, target value: 1
prediction: 2.58, target value: 1
prediction: -4.29, target value: 0
prediction: -6.53, target value: 0
prediction: -4.91, target value: 0
prediction: -4.59, target value: 0
prediction: 3.89, target value: 1
prediction: -6.02, target value: 0
prediction: 2.39, target value: 1
prediction: 2.69, target value: 1
prediction: -4.19, target value: 0
prediction: -4.33, target value: 0
prediction: -1.70, target value: 0
prediction: -4.05, target value: 0
prediction: 5.32, target value: 1
prediction: -7.32, target value: 0
prediction: 6.75, target value: 1
prediction: -0.94, target value: 1
prediction: -3.45, target value: 0
prediction: 2.64, target value: 1
prediction: -2.97, target value: 0
prediction: 6.28, target value: 1
prediction: 11.97, target value: 1
prediction: -2.69, target value: 0
prediction: -2.46, target value: 0
prediction: -0.05, target value: 0
prediction: 14.22, target value: 1
prediction: 5.20, target value: 1
prediction: -2.96, target value: 0
prediction: 5.14, target value: 1
prediction: 0.47, target value: 1
prediction: 6.07, target value: 1
prediction: -1.65, target value: 0
prediction: -0.09, target value: 0
prediction: -1.39, target value: 0
```

```
prediction: 0.18, target value: 1
prediction: -3.02, target value: 0
prediction: -2.11, target value: 0
prediction: 2.89, target value: 1
prediction: 6.52, target value: 1
prediction: -4.20, target value: 0
prediction: -5.75, target value: 0
prediction: -4.26, target value: 0
prediction: 0.43, target value: 1
prediction: 0.55, target value: 1
prediction: -7.52, target value: 0
prediction: -3.38, target value: 0
prediction: -3.53, target value: 0
prediction: -4.59, target value: 0
prediction: 1.98, target value: 1
prediction: -1.57, target value: 0
prediction: -3.47, target value: 0
prediction: 15.12, target value: 1
prediction: -2.54, target value: 0
prediction: -4.97, target value: 0
prediction: -2.27, target value: 0
prediction: -0.51, target value: 0
prediction: -4.49, target value: 0
prediction: -4.93, target value: 0
prediction: -2.75, target value: 0
prediction: -6.85, target value: 0
prediction: 3.70, target value: 1
prediction: 5.49, target value: 1
prediction: 2.57, target value: 1
prediction: -4.29, target value: 0
prediction: 4.59, target value: 1
prediction: 13.61, target value: 1
prediction: -1.84, target value: 0
prediction: -3.73, target value: 0
prediction: -3.43, target value: 0
prediction: 0.70, target value: 1
prediction: 3.40, target value: 1
prediction: -0.10, target value: 0
prediction: 6.78, target value: 1
prediction: -3.50, target value: 0
prediction: 2.53, target value: 1
prediction: 2.74, target value: 1
prediction: -0.82, target value: 0
prediction: 4.61, target value: 1
prediction: -1.75, target value: 1
prediction: -3.71, target value: 0
prediction: -4.36, target value: 0
prediction: 3.48, target value: 1
prediction: -4.36, target value: 0
prediction: -7.20, target value: 0
prediction: 1.95, target value: 1
prediction: -3.70, target value: 0
prediction: -2.41, target value: 0
prediction: -5.92, target value: 0
prediction: -4.18, target value: 0
prediction: 0.77, target value: 1
```

```
prediction: -1.65, target value: 0
prediction: -1.02, target value: 0
prediction: -3.54, target value: 0
prediction: -1.91, target value: 0
prediction: -3.86, target value: 0
prediction: -2.85, target value: 0
prediction: -5.30, target value: 0
prediction: -1.78, target value: 0
prediction: -3.21, target value: 0
prediction: 4.46, target value: 1
prediction: -0.22, target value: 0
prediction: -4.85, target value: 0
prediction: -5.94, target value: 0
prediction: -1.92, target value: 0
prediction: 4.14, target value: 1
prediction: 8.84, target value: 1
prediction: -2.63, target value: 0
prediction: 8.96, target value: 1
prediction: -2.70, target value: 0
prediction: -6.46, target value: 0
prediction: 2.02, target value: 1
prediction: 6.87, target value: 1
prediction: -1.99, target value: 0
prediction: -3.82, target value: 0
prediction: -0.02, target value: 1
prediction: 2.31, target value: 1
prediction: -6.32, target value: 0
prediction: -6.19, target value: 0
prediction: -7.86, target value: 0
prediction: -4.66, target value: 0
prediction: 2.99, target value: 1
prediction: -4.85, target value: 0
prediction: -5.52, target value: 0
prediction: 14.99, target value: 1
prediction: 11.46, target value: 1
prediction: 2.31, target value: 1
prediction: -5.49, target value: 0
prediction: 0.28, target value: 1
prediction: -5.18, target value: 0
prediction: 2.37, target value: 1
prediction: -3.68, target value: 0
prediction: -4.04, target value: 0
prediction: -4.56, target value: 0
prediction: 3.37, target value: 1
prediction: -3.18, target value: 0
prediction: -7.76, target value: 0
prediction: 1.84, target value: 1
prediction: 1.83, target value: 1
prediction: -3.60, target value: 0
prediction: 2.51, target value: 1
prediction: 1.21, target value: 1
prediction: 5.58, target value: 1
prediction: 2.52, target value: 1
prediction: -1.79, target value: 0
prediction: 2.96, target value: 1
prediction: 11.49, target value: 1
```

```
prediction: 6.17, target value: 1
prediction: -1.42, target value: 0
prediction: 0.01, target value: 1
prediction: -4.90, target value: 0
prediction: 1.70, target value: 1
prediction: -0.37, target value: 0
prediction: -2.23, target value: 0
prediction: 6.77, target value: 1
prediction: -3.89, target value: 0
prediction: 17.05, target value: 1
prediction: 0.76, target value: 1
prediction: 2.74, target value: 1
prediction: 0.44, target value: 1
prediction: -2.15, target value: 0
prediction: -5.19, target value: 0
prediction: 7.43, target value: 1
prediction: 8.82, target value: 1
prediction: -3.51, target value: 0
prediction: -2.09, target value: 0
prediction: -4.73, target value: 0
prediction: 2.89, target value: 1
prediction: -2.76, target value: 0
prediction: -1.02, target value: 0
prediction: -5.47, target value: 0
prediction: -1.19, target value: 0
prediction: -1.62, target value: 0
prediction: 2.12, target value: 1
prediction: 4.53, target value: 1
prediction: -4.25, target value: 0
prediction: -2.99, target value: 0
prediction: 6.44, target value: 1
prediction: -5.85, target value: 0
prediction: -2.25, target value: 0
prediction: 13.01, target value: 1
prediction: 4.56, target value: 1
prediction: -0.35, target value: 0
prediction: 6.60, target value: 1
prediction: -2.79, target value: 0
prediction: -5.30, target value: 0
prediction: -1.68, target value: 0
prediction: -2.28, target value: 0
prediction: 5.30, target value: 1
prediction: -3.41, target value: 0
prediction: -3.82, target value: 0
prediction: -1.58, target value: 0
prediction: -2.59, target value: 0
prediction: -3.72, target value: 0
prediction: 9.14, target value: 1
prediction: -3.82, target value: 0
prediction: 8.16, target value: 1
prediction: 2.59, target value: 1
prediction: 6.29, target value: 1
prediction: 0.11, target value: 1
prediction: 8.84, target value: 1
prediction: 4.14, target value: 1
prediction: 8.56, target value: 1
```

```
prediction: 6.28, target value: 1
prediction: 7.09, target value: 1
prediction: 0.89, target value: 1
prediction: 3.55, target value: 1
prediction: -0.82, target value: 1
prediction: 3.98, target value: 1
prediction: 12.62, target value: 1
prediction: -3.48, target value: 0
prediction: -2.54, target value: 0
prediction: -3.05, target value: 0
prediction: -3.69, target value: 0
prediction: -4.88, target value: 0
prediction: -4.79, target value: 0
prediction: 10.23, target value: 1
prediction: -5.41, target value: 0
prediction: 2.43, target value: 1
prediction: -2.48, target value: 0
prediction: -5.63, target value: 0
prediction: 1.46, target value: 1
prediction: -3.43, target value: 0
prediction: -2.36, target value: 0
prediction: 7.90, target value: 1
prediction: -3.94, target value: 0
prediction: 6.03, target value: 1
prediction: 2.53, target value: 1
prediction: -3.63, target value: 0
prediction: -4.96, target value: 0
prediction: -2.93, target value: 0
prediction: -5.69, target value: 0
prediction: -3.35, target value: 0
prediction: -4.11, target value: 0
prediction: -2.04, target value: 0
prediction: -0.22, target value: 0
prediction: -2.72, target value: 0
prediction: -3.36, target value: 0
prediction: -4.51, target value: 0
prediction: -4.18, target value: 0
prediction: -7.29, target value: 0
prediction: -3.30, target value: 1
prediction: -2.77, target value: 0
prediction: -5.08, target value: 0
prediction: 8.58, target value: 1
prediction: -3.28, target value: 0
prediction: 8.32, target value: 1
prediction: -5.11, target value: 0
prediction: -4.67, target value: 0
prediction: -3.73, target value: 0
prediction: -4.60, target value: 0
prediction: -7.65, target value: 0
prediction: -5.44, target value: 0
prediction: -4.79, target value: 0
prediction: -3.99, target value: 0
prediction: -3.15, target value: 0
prediction: -3.93, target value: 0
prediction: -5.89, target value: 0
prediction: -6.80, target value: 0
```

```
prediction: -5.64, target value: 0
prediction: -6.06, target value: 0
prediction: 3.56, target value: 1
prediction: -3.12, target value: 0
prediction: -5.59, target value: 0
prediction: -4.45, target value: 0
prediction: 3.95, target value: 1
prediction: -2.90, target value: 0
prediction: 10.35, target value: 1
prediction: -4.03, target value: 0
prediction: -3.59, target value: 0
prediction: -3.83, target value: 0
prediction: -5.54, target value: 0
prediction: 3.39, target value: 1
prediction: 1.48, target value: 1
prediction: 1.90, target value: 1
prediction: -1.83, target value: 0
prediction: -4.12, target value: 0
prediction: -5.48, target value: 0
prediction: -4.33, target value: 0
prediction: 5.04, target value: 1
prediction: -4.83, target value: 0
prediction: 6.67, target value: 1
prediction: -4.35, target value: 0
prediction: 12.11, target value: 1
prediction: -0.42, target value: 0
prediction: -4.46, target value: 0
prediction: -3.82, target value: 0
prediction: 7.09, target value: 1
prediction: -3.52, target value: 0
prediction: -5.87, target value: 0
prediction: -3.89, target value: 0
prediction: -1.52, target value: 0
prediction: -4.40, target value: 0
prediction: -3.85, target value: 0
prediction: -5.06, target value: 0
prediction: 4.31, target value: 1
prediction: 15.27, target value: 1
prediction: 2.92, target value: 1
prediction: -5.88, target value: 0
prediction: -3.12, target value: 0
prediction: -1.37, target value: 0
prediction: -3.46, target value: 0
prediction: -6.00, target value: 0
prediction: -4.53, target value: 0
prediction: -5.05, target value: 0
prediction: -2.67, target value: 0
prediction: -3.06, target value: 0
prediction: -0.20, target value: 0
prediction: -3.60, target value: 0
prediction: 5.23, target value: 1
prediction: 8.20, target value: 1
prediction: -2.58, target value: 0
prediction: 9.84, target value: 1
prediction: 10.43, target value: 1
prediction: 5.14, target value: 1
```

```
prediction: -3.18, target value: 0
prediction: 4.64, target value: 1
prediction: 6.27, target value: 1
prediction: -3.20, target value: 0
prediction: -0.76, target value: 0
prediction: -3.26, target value: 0
prediction: -2.28, target value: 0
prediction: -2.78, target value: 0
prediction: 2.89, target value: 1
prediction: -2.41, target value: 0
prediction: -4.82, target value: 0
prediction: -3.91, target value: 0
prediction: -2.30, target value: 0
prediction: -3.41, target value: 0
prediction: 0.04, target value: 1
prediction: -4.11, target value: 0
prediction: -4.17, target value: 0
prediction: -5.01, target value: 0
prediction: 5.07, target value: 1
prediction: -5.61, target value: 0
prediction: -5.95, target value: 0
prediction: 5.07, target value: 1
prediction: 10.29, target value: 1
prediction: -2.62, target value: 0
prediction: -3.26, target value: 0
prediction: -0.78, target value: 0
prediction: -3.66, target value: 0
prediction: -5.34, target value: 0
prediction: -3.63, target value: 0
prediction: 7.13, target value: 1
prediction: -4.30, target value: 0
prediction: -3.39, target value: 0
prediction: -3.25, target value: 0
prediction: -4.95, target value: 0
prediction: -3.61, target value: 0
prediction: -1.05, target value: 0
prediction: -3.31, target value: 0
prediction: 4.04, target value: 1
prediction: -2.37, target value: 0
prediction: -2.75, target value: 0
prediction: -3.76, target value: 0
prediction: -6.17, target value: 0
prediction: 0.22, target value: 0
prediction: 0.56, target value: 1
prediction: -2.71, target value: 0
prediction: -3.96, target value: 0
prediction: 7.23, target value: 1
prediction: -4.39, target value: 0
prediction: -3.90, target value: 0
prediction: -3.18, target value: 0
prediction: -0.17, target value: 0
prediction: -3.03, target value: 0
prediction: -1.18, target value: 0
prediction: -3.58, target value: 0
prediction: -5.70, target value: 0
prediction: -3.80, target value: 0
```

```
prediction: -2.91, target value: 0
prediction: -6.19, target value: 0
prediction: -4.59, target value: 0
prediction: 3.91, target value: 1
prediction: -3.17, target value: 0
prediction: 6.00, target value: 1
prediction: 5.77, target value: 1
prediction: -2.46, target value: 0
prediction: 1.28, target value: 1
prediction: -2.66, target value: 0
prediction: -2.60, target value: 0
prediction: -2.56, target value: 0
prediction: -3.87, target value: 0
prediction: -2.32, target value: 0
prediction: 3.87, target value: 1
prediction: -4.35, target value: 0
prediction: -5.85, target value: 0
prediction: 1.88, target value: 1
prediction: -2.09, target value: 0
prediction: 5.81, target value: 1
prediction: -1.55, target value: 0
prediction: -1.25, target value: 0
prediction: 7.46, target value: 1
prediction: -3.98, target value: 0
prediction: 4.71, target value: 1
prediction: -2.53, target value: 0
prediction: -1.96, target value: 0
prediction: -3.15, target value: 0
prediction: -0.89, target value: 0
prediction: -1.27, target value: 0
prediction: -2.20, target value: 0
prediction: -2.64, target value: 0
prediction: -4.46, target value: 0
prediction: 5.84, target value: 1
prediction: 23.27, target value: 1
prediction: -2.02, target value: 0
prediction: -3.77, target value: 0
prediction: -2.93, target value: 0
prediction: -0.63, target value: 0
prediction: -1.07, target value: 0
prediction: -5.18, target value: 0
prediction: 5.18, target value: 1
prediction: -1.02, target value: 0
prediction: -4.32, target value: 0
prediction: -2.07, target value: 0
prediction: -1.77, target value: 0
prediction: -3.30, target value: 0
prediction: -4.25, target value: 0
prediction: -2.91, target value: 0
prediction: -0.79, target value: 0
prediction: -3.65, target value: 0
prediction: -3.97, target value: 0
prediction: 2.56, target value: 1
prediction: -3.69, target value: 0
prediction: -2.31, target value: 0
prediction: -2.19, target value: 0
```

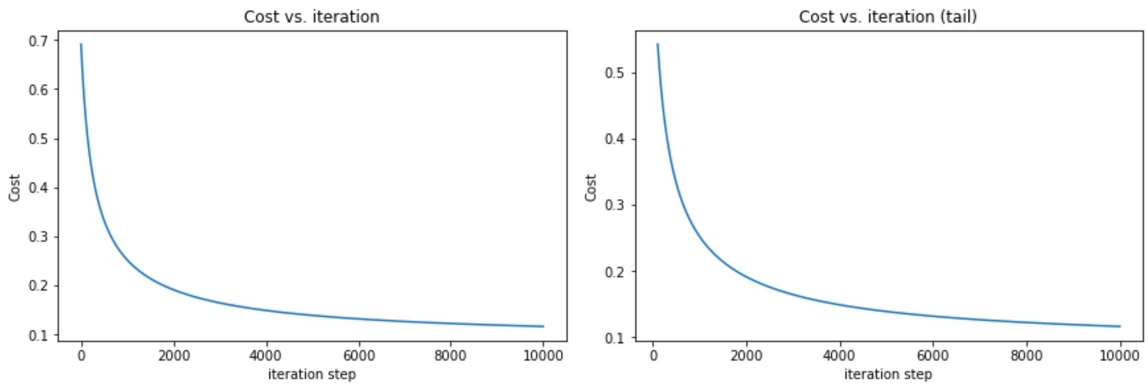
```
prediction: -2.55, target value: 0
prediction: -0.74, target value: 0
prediction: -1.56, target value: 0
prediction: -2.05, target value: 0
prediction: 6.93, target value: 1
prediction: -2.80, target value: 0
prediction: 0.56, target value: 1
prediction: -2.72, target value: 0
prediction: -1.02, target value: 0
prediction: 4.16, target value: 1
prediction: -6.10, target value: 0
prediction: -3.18, target value: 0
prediction: -1.28, target value: 0
prediction: -1.14, target value: 0
prediction: -2.94, target value: 0
prediction: 4.40, target value: 1
prediction: 7.33, target value: 1
prediction: -1.35, target value: 0
prediction: 2.42, target value: 1
prediction: -2.57, target value: 0
prediction: 11.00, target value: 1
prediction: -3.30, target value: 0
prediction: -3.56, target value: 0
prediction: -2.68, target value: 0
prediction: -3.93, target value: 0
prediction: -1.30, target value: 0
prediction: 3.98, target value: 1
prediction: -4.84, target value: 0
prediction: -3.27, target value: 0
prediction: 2.28, target value: 1
prediction: -1.62, target value: 0
prediction: -0.25, target value: 1
prediction: -3.26, target value: 0
prediction: 4.51, target value: 1
prediction: 5.18, target value: 1
prediction: -1.17, target value: 0
prediction: -2.14, target value: 0
prediction: -4.05, target value: 0
prediction: 12.13, target value: 1
prediction: -5.36, target value: 0
prediction: -1.32, target value: 0
prediction: -5.08, target value: 0
prediction: -5.75, target value: 0
prediction: -0.62, target value: 0
prediction: -3.84, target value: 0
prediction: -1.85, target value: 0
prediction: -3.85, target value: 0
prediction: -2.34, target value: 0
prediction: -2.17, target value: 0
prediction: -2.92, target value: 0
prediction: 5.93, target value: 1
prediction: -3.96, target value: 0
prediction: 7.30, target value: 1
prediction: 0.63, target value: 1
prediction: -0.82, target value: 0
prediction: -6.00, target value: 0
```

```
prediction: -4.71, target value: 0
prediction: -4.57, target value: 0
prediction: 0.60, target value: 0
prediction: -0.68, target value: 0
prediction: -1.76, target value: 0
prediction: -2.55, target value: 0
prediction: -1.94, target value: 0
prediction: -5.69, target value: 0
prediction: -5.09, target value: 0
prediction: -5.48, target value: 0
prediction: -3.07, target value: 0
prediction: -5.79, target value: 0
prediction: -3.68, target value: 0
prediction: -2.30, target value: 0
prediction: -5.99, target value: 0
prediction: -1.98, target value: 0
prediction: -3.27, target value: 0
prediction: -5.69, target value: 0
prediction: -4.68, target value: 0
prediction: -1.33, target value: 0
prediction: -2.15, target value: 0
prediction: -0.69, target value: 0
prediction: -4.73, target value: 0
prediction: 6.54, target value: 1
prediction: 10.51, target value: 1
prediction: 9.27, target value: 1
```

Evaluation

Here we measure both the accuracy of our model and the cost over the gradient descent iterations, so we can see how much it learned per iteration, as well as how accurate it is in the end.

In [9]:
```
# plot cost versus iteration
fig, (ax1, ax2) = plt.subplots(1, 2, constrained_layout=True, figsiz
e=(12, 4))
ax1.plot(J_hist)
ax2.plot(100 + np.arange(len(J_hist[100:])), J_hist[100:])
ax1.set_title("Cost vs. iteration");  ax2.set_title("Cost vs. iterat
ion (tail)")
ax1.set_ylabel('Cost')             ;  ax2.set_ylabel('Cost')
ax1.set_xlabel('iteration step')   ;  ax2.set_xlabel('iteration step
')
plt.show()
```



In [10]:
```
def predict(X, w, b):
    """
    Predict whether the label is 0 or 1 using learned logistic
    regression parameters w

    Args:
    X : (ndarray Shape (m, n))
    w : (array_like Shape (n,))      Parameters of the model
    b : (scalar, float)              Parameter of the model

    Returns:
    p: (ndarray (m,1))
        The predictions for X using a threshold at 0.5
    """
    # number of training examples
    m, n = X.shape
    p = np.zeros(m)

    p = np.asarray([1 if sigmoid(np.dot(X[i], w)+b) >= 0.5 else 0 fo
r i in range(m)])
    return p
```

In [11]:
```
p = predict(X_norm, w_final,b_final)
print('Train Accuracy: %f'%(np.mean(p == y_train) * 100))
```

```
Train Accuracy: 98.066784
```