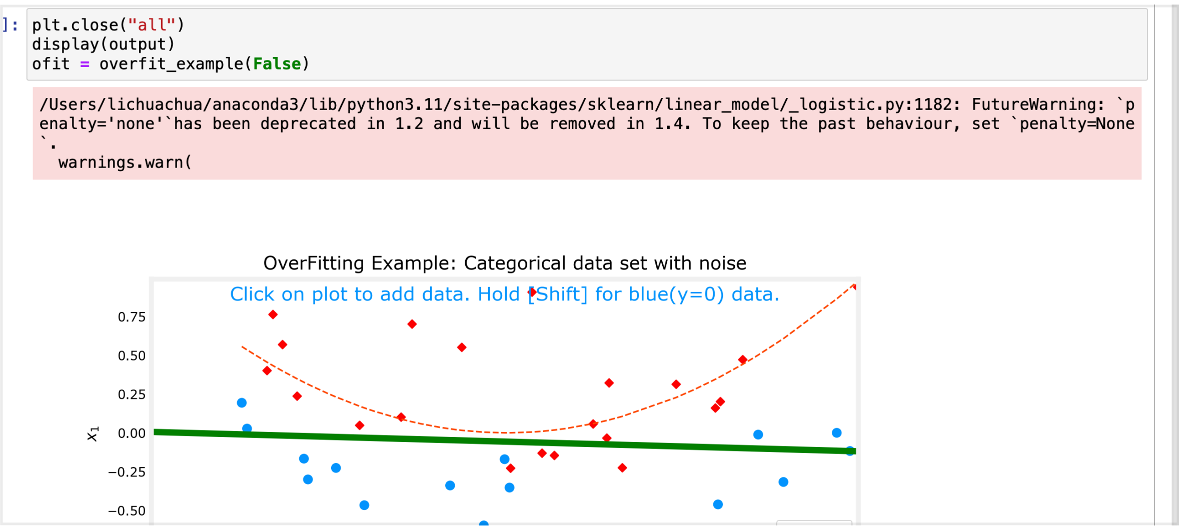
机器学习 C1-Lab解决问题

C1_W1_Lab04_Cost_function_Soln

需要安装pip install ipympl

C1_W3_Lab08_Overfitting_Soln

选择Categorical报warning：

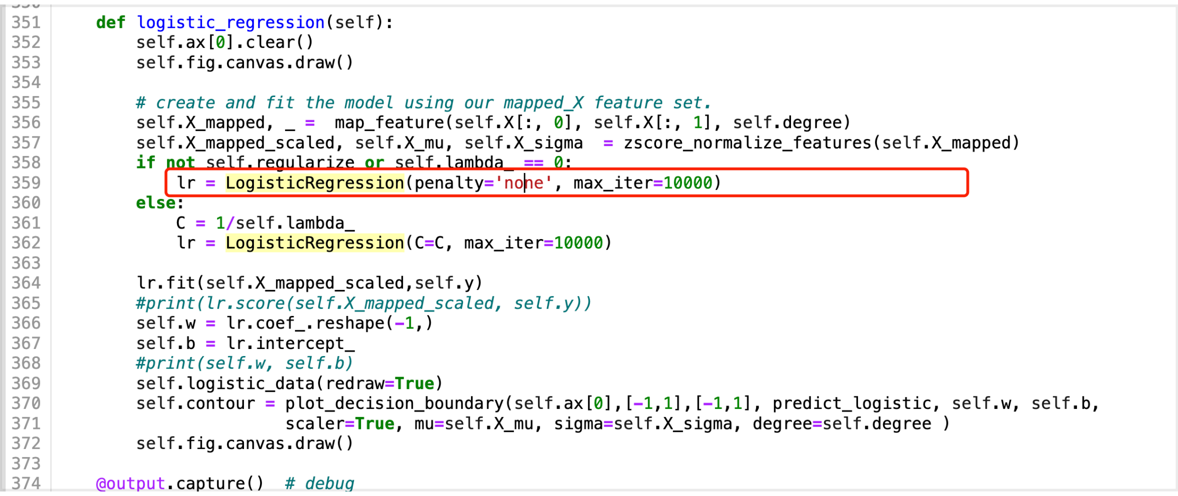


```
plt.close("all")
display(output)
ofit = overfit_example(False)
```

```
/Users/lichuachua/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:1182: FutureWarning: `p
enalty='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep the past behaviour, set `penalty=None
`.
  warnings.warn(
```

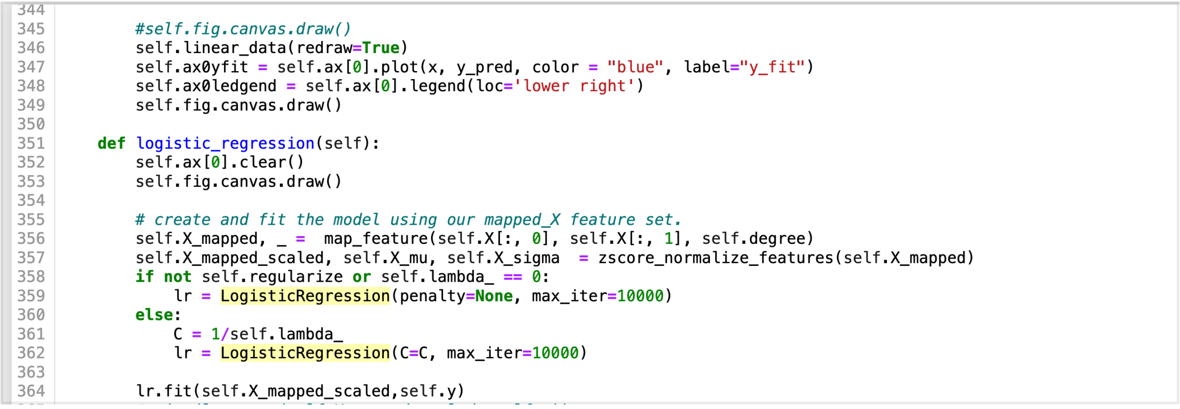
OverFitting Example: Categorical data set with noise
Click on plot to add data. Hold [Shift] for blue(y=0) data.
0.75
0.50
0.25
0.00
−0.25
−0.50
x_1

解决：
更改这行错误



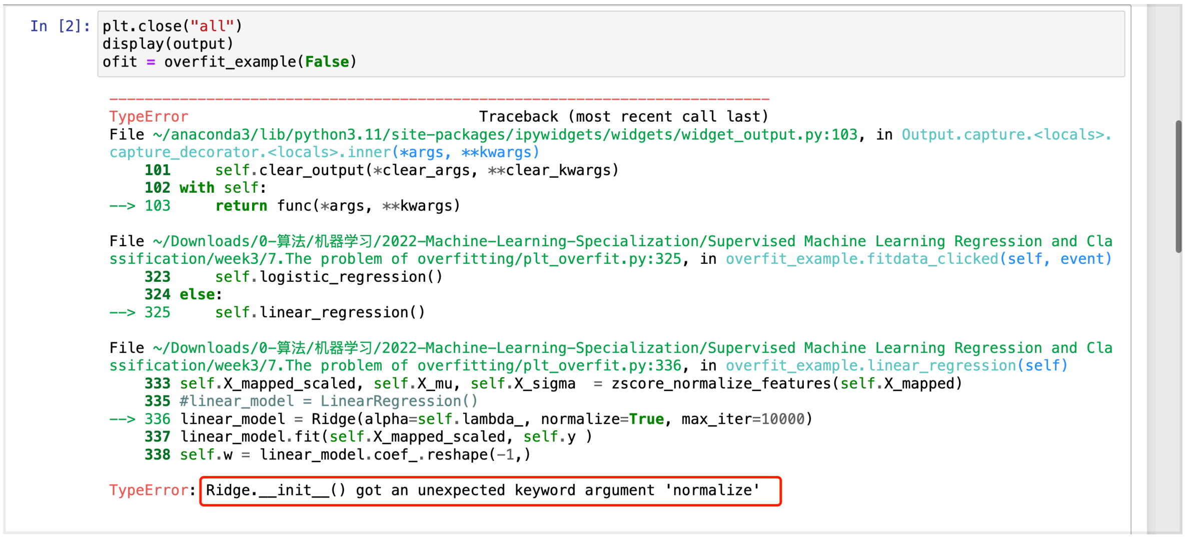
```
    def logistic_regression(self):
        self.ax[0].clear()
        self.fig.canvas.draw()

        # create and fit the model using our mapped_X feature set.
        self.X_mapped, _ =  map_feature(self.X[:, 0], self.X[:, 1], self.degree)
        self.X_mapped_scaled, self.X_mu, self.X_sigma  = zscore_normalize_features(self.X_mapped)
        if not self.regularize or self.lambda_ == 0:
            lr = LogisticRegression(penalty='none', max_iter=10000)
        else:
            C = 1/self.lambda_
            lr = LogisticRegression(C=C, max_iter=10000)

        lr.fit(self.X_mapped_scaled,self.y)
        #print(lr.score(self.X_mapped_scaled, self.y))
        self.w = lr.coef_.reshape(-1,)
        self.b = lr.intercept_
        #print(self.w, self.b)
        self.logistic_data(redraw=True)
        self.contour = plot_decision_boundary(self.ax[0],[-1,1],[-1,1], predict_logistic, self.w, self.b,
                       scaler=True, mu=self.X_mu, sigma=self.X_sigma, degree=self.degree )
        self.fig.canvas.draw()

    @output.capture()  # debug
```

改为None



```
        #self.fig.canvas.draw()
        self.linear_data(redraw=True)
        self.ax0yfit = self.ax[0].plot(x, y_pred, color = "blue", label="y_fit")
        self.ax0ledgend = self.ax[0].legend(loc='lower right')
        self.fig.canvas.draw()

    def logistic_regression(self):
        self.ax[0].clear()
        self.fig.canvas.draw()

        # create and fit the model using our mapped_X feature set.
        self.X_mapped, _ =  map_feature(self.X[:, 0], self.X[:, 1], self.degree)
        self.X_mapped_scaled, self.X_mu, self.X_sigma  = zscore_normalize_features(self.X_mapped)
        if not self.regularize or self.lambda_ == 0:
            lr = LogisticRegression(penalty=None, max_iter=10000)
        else:
            C = 1/self.lambda_
            lr = LogisticRegression(C=C, max_iter=10000)

        lr.fit(self.X_mapped_scaled,self.y)
```

选择Regression时会报错：

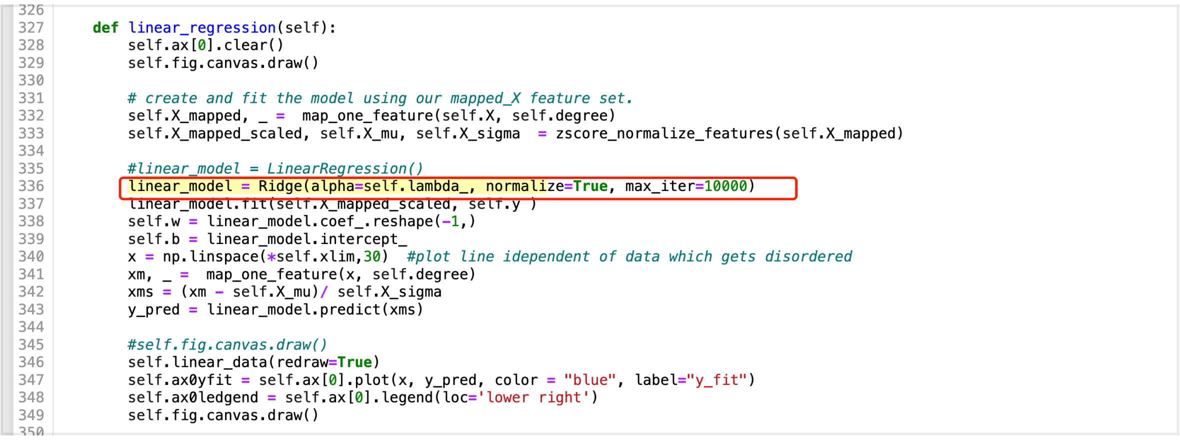


```
In [2]: plt.close("all")
        display(output)
        ofit = overfit_example(False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File ~/anaconda3/lib/python3.11/site-packages/ipywidgets/widgets/widget_output.py:103, in Output.capture.<locals>.
capture_decorator.<locals>.inner(*args, **kwargs)
    101     self.clear_output(*clear_args, **clear_kwargs)
    102 with self:
--> 103     return func(*args, **kwargs)

File ~/Downloads/0-算法/机器学习/2022-Machine-Learning-Specialization/Supervised Machine Learning Regression and Cla
ssification/week3/7.The problem of overfitting/plt_overfit.py:325, in overfit_example.fitdata_clicked(self, event)
    323     self.logistic_regression()
    324 else:
--> 325     self.linear_regression()

File ~/Downloads/0-算法/机器学习/2022-Machine-Learning-Specialization/Supervised Machine Learning Regression and Cla
ssification/week3/7.The problem of overfitting/plt_overfit.py:336, in overfit_example.linear_regression(self)
    333 self.X_mapped_scaled, self.X_mu, self.X_sigma  = zscore_normalize_features(self.X_mapped)
    335 #linear_model = LinearRegression()
--> 336 linear_model = Ridge(alpha=self.lambda_, normalize=True, max_iter=10000)
    337 linear_model.fit(self.X_mapped_scaled, self.y )
    338 self.w = linear_model.coef_.reshape(-1,)

TypeError: Ridge.__init__() got an unexpected keyword argument 'normalize'
```

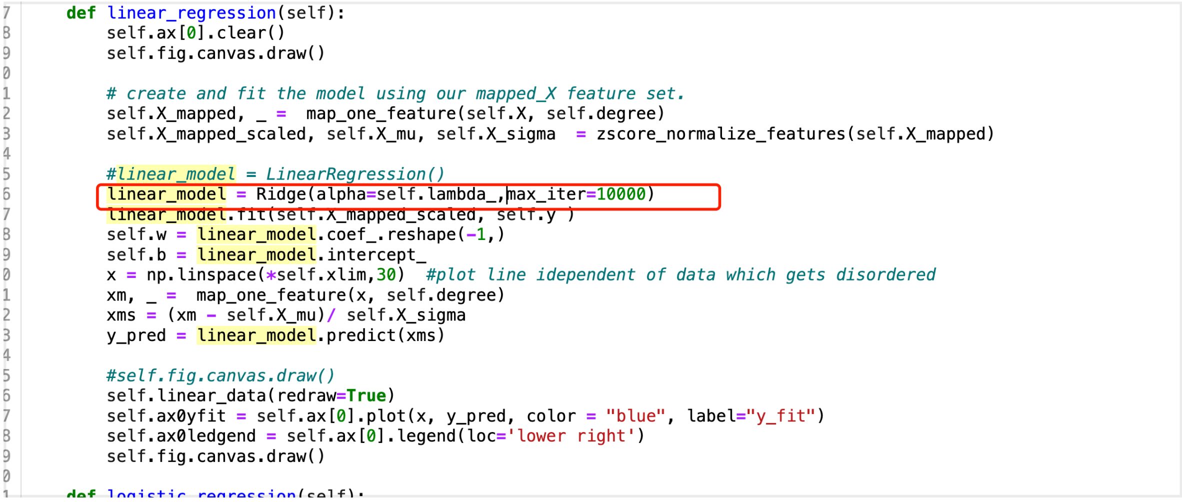
根据 scikit-learn 文档，在 scikit-learn 版本 0.24.0 中，normalize 参数被弃用，并计划在 scikit-learn 版本 1.0 中删除。

更改此行：



```

    def linear_regression(self):
        self.ax[0].clear()
        self.fig.canvas.draw()

        # create and fit the model using our mapped_X feature set.
        self.X_mapped, _ =  map_one_feature(self.X, self.degree)
        self.X_mapped_scaled, self.X_mu, self.X_sigma  = zscore_normalize_features(self.X_mapped)

        #linear_model = LinearRegression()
        linear_model = Ridge(alpha=self.lambda_, normalize=True, max_iter=10000)
        linear_model.fit(self.X_mapped_scaled, self.y )
        self.w = linear_model.coef_.reshape(-1,)
        self.b = linear_model.intercept_
        x = np.linspace(*self.xlim,30)  #plot line independent of data which gets disordered
        xm, _ =  map_one_feature(x, self.degree)
        xms = (xm - self.X_mu)/ self.X_sigma
        y_pred = linear_model.predict(xms)

        #self.fig.canvas.draw()
        self.linear_data(redraw=True)
        self.ax0yfit = self.ax[0].plot(x, y_pred, color = "blue", label="y_fit")
        self.ax0ledgend = self.ax[0].legend(loc='lower right')
        self.fig.canvas.draw()
```

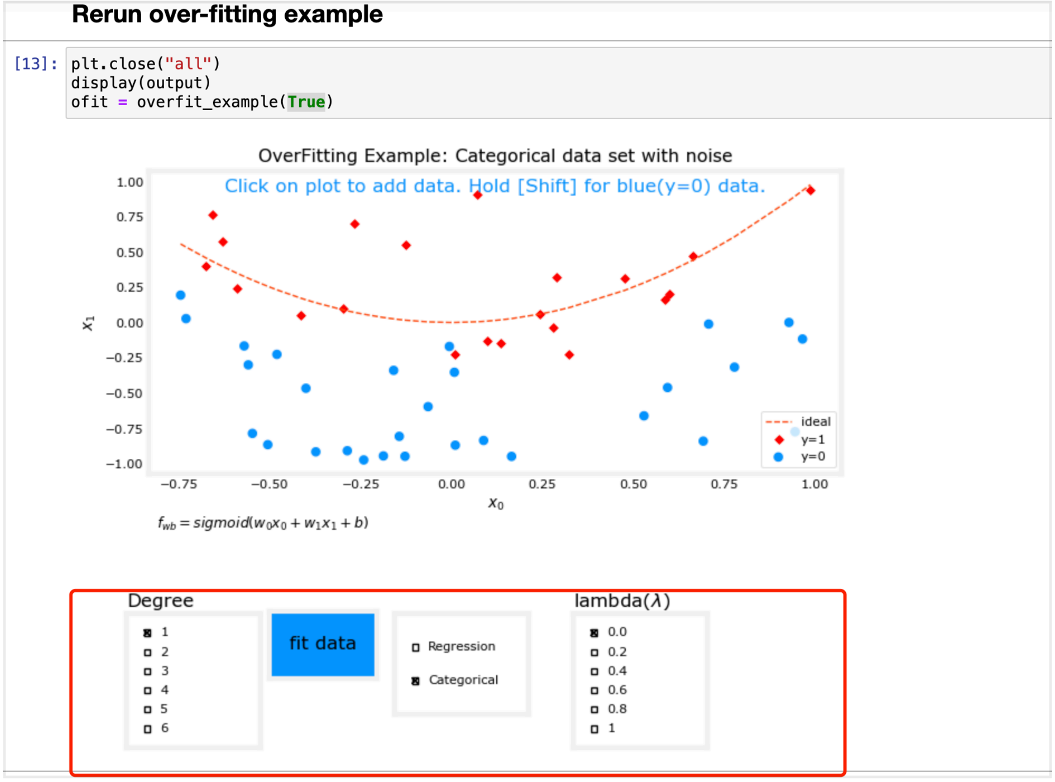
去掉归一化（省事儿）



```
    def linear_regression(self):
        self.ax[0].clear()
        self.fig.canvas.draw()

        # create and fit the model using our mapped_X feature set.
        self.X_mapped, _ =  map_one_feature(self.X, self.degree)
        self.X_mapped_scaled, self.X_mu, self.X_sigma  = zscore_normalize_features(self.X_mapped)

        #linear_model = LinearRegression()
        linear_model = Ridge(alpha=self.lambda_,max_iter=10000)
        linear_model.fit(self.X_mapped_scaled, self.y )
        self.w = linear_model.coef_.reshape(-1,)
        self.b = linear_model.intercept_
        x = np.linspace(*self.xlim,30)  #plot line idependent of data which gets disordered
        xm, _ =  map_one_feature(x, self.degree)
        xms = (xm - self.X_mu)/ self.X_sigma
        y_pred = linear_model.predict(xms)

        #self.fig.canvas.draw()
        self.linear_data(redraw=True)
        self.ax0yfit = self.ax[0].plot(x, y_pred, color = "blue", label="y_fit")
        self.ax0ledgend = self.ax[0].legend(loc='lower right')
        self.fig.canvas.draw()

    def logistic_regression(self):
```

C1_W3_Lab09_Regularization_Soln

灰色无法点击：



Rerun over-fitting example

```
[13]: plt.close("all")
      display(output)
      ofit = overfit_example(True)
```

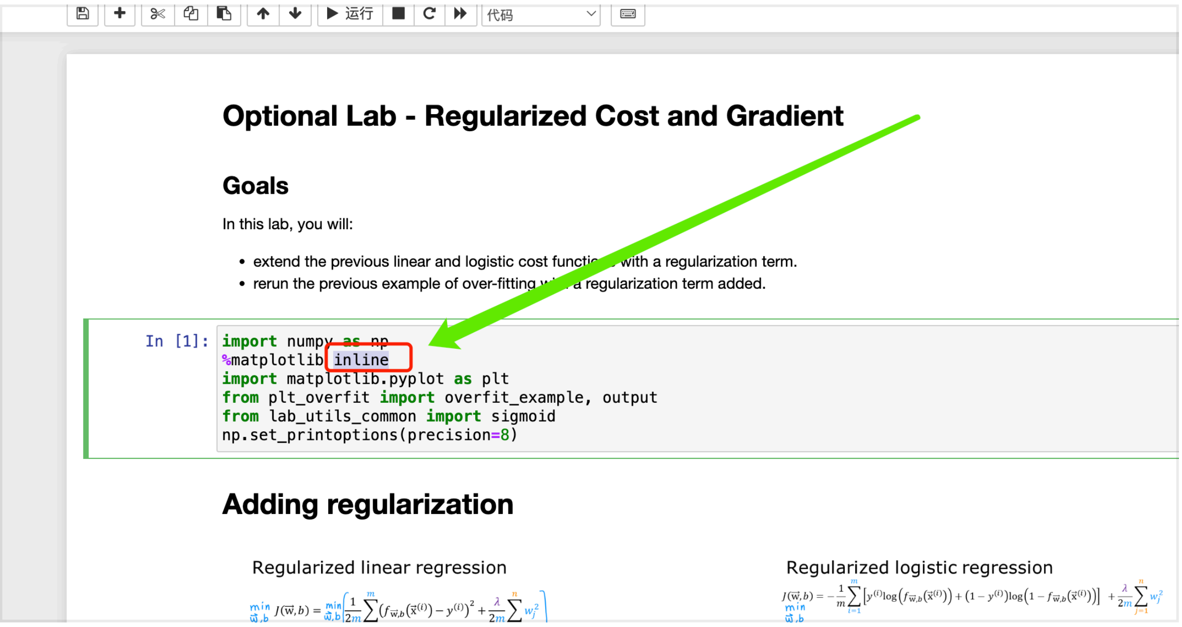
OverFitting Example: Categorical data set with noise
Click on plot to add data. Hold [Shift] for blue(y=0) data.
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
x_1
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x_0
ideal
y=1
y=0
$f_{wb} = sigmoid(w_0x_0 + w_1x_1 + b)$
fit data

Degree
☒ 1
□ 2
□ 3
□ 4
□ 5
□ 6

□ Regression
☒ Categorical

lambda(λ)
☒ 0.0
□ 0.2
□ 0.4
□ 0.6
□ 0.8
□ 1

解决：
更改mtplotlib的形式



Optional Lab - Regularized Cost and Gradient

Goals

In this lab, you will:

- extend the previous linear and logistic cost functions with a regularization term.
- rerun the previous example of over-fitting with a regularization term added.

```
In [1]: import numpy as np
        %matplotlib inline
        import matplotlib.pyplot as plt
        from plt_overfit import overfit_example, output
        from lab_utils_common import sigmoid
        np.set_printoptions(precision=8)
```

Adding regularization

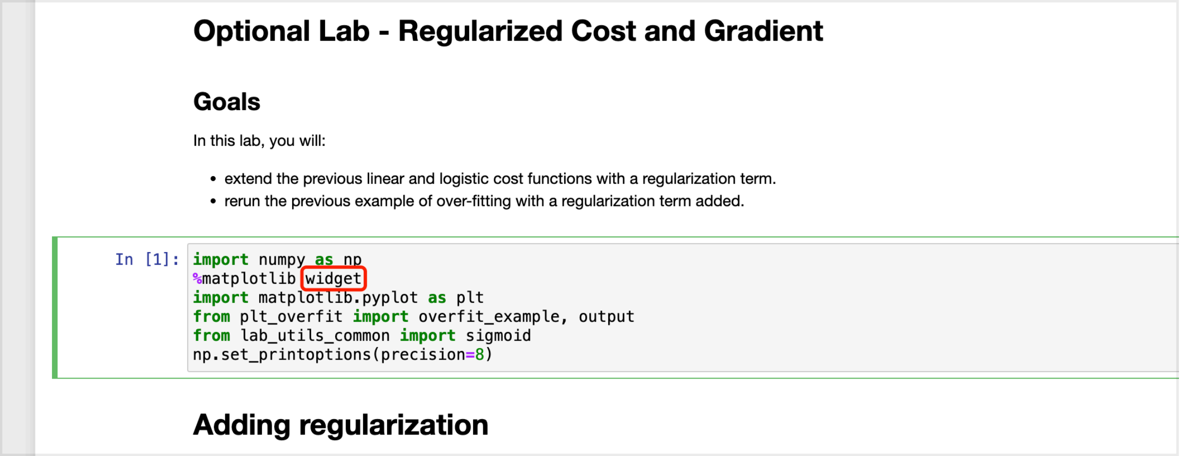
Regularized linear regression

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

Regularized logistic regression

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

改为：



Optional Lab - Regularized Cost and Gradient

Goals

In this lab, you will:

- extend the previous linear and logistic cost functions with a regularization term.
- rerun the previous example of over-fitting with a regularization term added.

```
In [1]: import numpy as np
        %matplotlib widget
        import matplotlib.pyplot as plt
        from plt_overfit import overfit_example, output
        from lab_utils_common import sigmoid
        np.set_printoptions(precision=8)
```

Adding regularization