Poverty Prediction

Shao Hung Lin

Chendong Cai

Background

- Competition platform: DrivenData
- Data source: World Bank
- Purpose of the project: build a model to accurately predict the poverty status using various survey data

Data Summary (household)

- Household data: 8203 observations, 346 features (4 numerical features)

	wBXbHZmp	SlDKnCuu	AlDbXTlZ	...	poor
id					
80389	JhtDR	GUusz	aQeIm	...	True
9370	JhtDR	GUusz	cecIq	...	True
39883	JhtDR	GUusz	aQeIm	...	False
18327	JhtDR	alLXR	cecIq	...	True
88416	JhtDR	GUusz	cecIq	...	True

Data Summary (individual)

- Individual data: 37560 observations, 44 features (1 numerical feature)

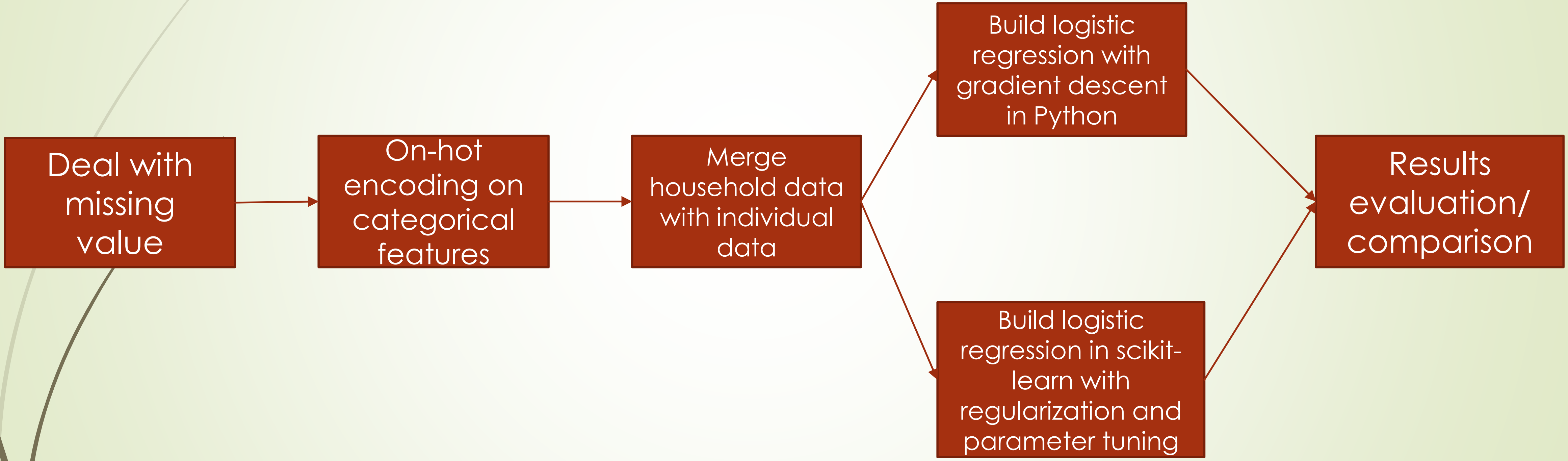
		HeUgMnzF	CaukPfUC	xqUooaNJ	...	poor
id	**iid**					
80389	**1**	XJsPz	mOlYV	dSJoN	...	True
	2	XJsPz	mOlYV	JTCKs	...	True
	3	TRFel	mOlYV	JTCKs	...	True
	4	XJsPz	yAyAe	JTCKs	...	True
9370	**1**	XJsPz	mOlYV	JTCKs	...	True

Performance metric: mean log loss

$$\text{MeanLogLoss} = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log \widehat{y_n} + (1 - y_n)\log(1 - \widehat{y_n})]$$

id	Poor predicted probability
418	0.32
41249	0.28
16205	0.58
97051	0.36
67756	0.63

Workflow


Deal with missing value
On-hot encoding on categorical features
Merge household data with individual data
Build logistic regression with gradient descent in Python
Build logistic regression in scikit-learn with regularization and parameter tuning
Results evaluation/ comparison

Data Preprocessing

- Dealing with missing values

	Missing counts	Missing percentage
OdXpbPGJ	6268	16.69%

	Count	Percentage
4	29436	78.37%

	Missing counts	Missing percentage
id	0	0.0%
HeUgMnzF	0	0.0%
CaukPfUC	0	0.0%
MzEtIdUF	0	0.0%
gtnNTNam	0	0.0%
SWoXNmPc	0	0.0%
eXbOkwhl	0	0.0%
OdXpbPGJ	6268	16.69%
XONDGWjH	0	0.0%
KsFoQcUV	0	0.0%
qYRZCuJD	0	0.0%
FPQrjGnS	0	0.0%
hOamrctW	0	0.0%
XacGrSou	0	0.0%
UsmeXdIS	0	0.0%
igHwZsYz	0	0.0%
cxWuAOZv	0	0.0%
AQpdiRUz	0	0.0%
AoLwmlEH	0	0.0%
nLUXHpZr	0	0.0%
CRLISiFu	0	0.0%
iYpOAjPW	0	0.0%

Data Preprocessing

- One-hot encoding for categorical features

	A	B	C	D	E	F	G	H	I
1	Original data:			One-hot encoding format:					
2	id	Color		id	White	Red	Black	Purple	Gold
3	1	White		1	1	0	0	0	0
4	2	Red		2	0	1	0	0	0
5	3	Black		3	0	0	1	0	0
6	4	Purple		4	0	0	0	1	0
7	5	Gold		5	0	0	0	0	1
8									

Data Preprocessing

- Merge household data with individual data

Individual data

id	iid	CaukPfUC_kzSFB	CaukPfUC_mOIYV	CaukPfUC_yAyAe	MzEtIdUF_FRcdT	MzEtIdUF_UFoKR	MzEtIdUF_axSTs
14	2	0	1	0	0	0	1
14	1	0	1	0	0	1	0
18	3	0	1	0	0	0	1
18	1	1	0	0	0	0	1
18	2	0	1	0	0	0	1

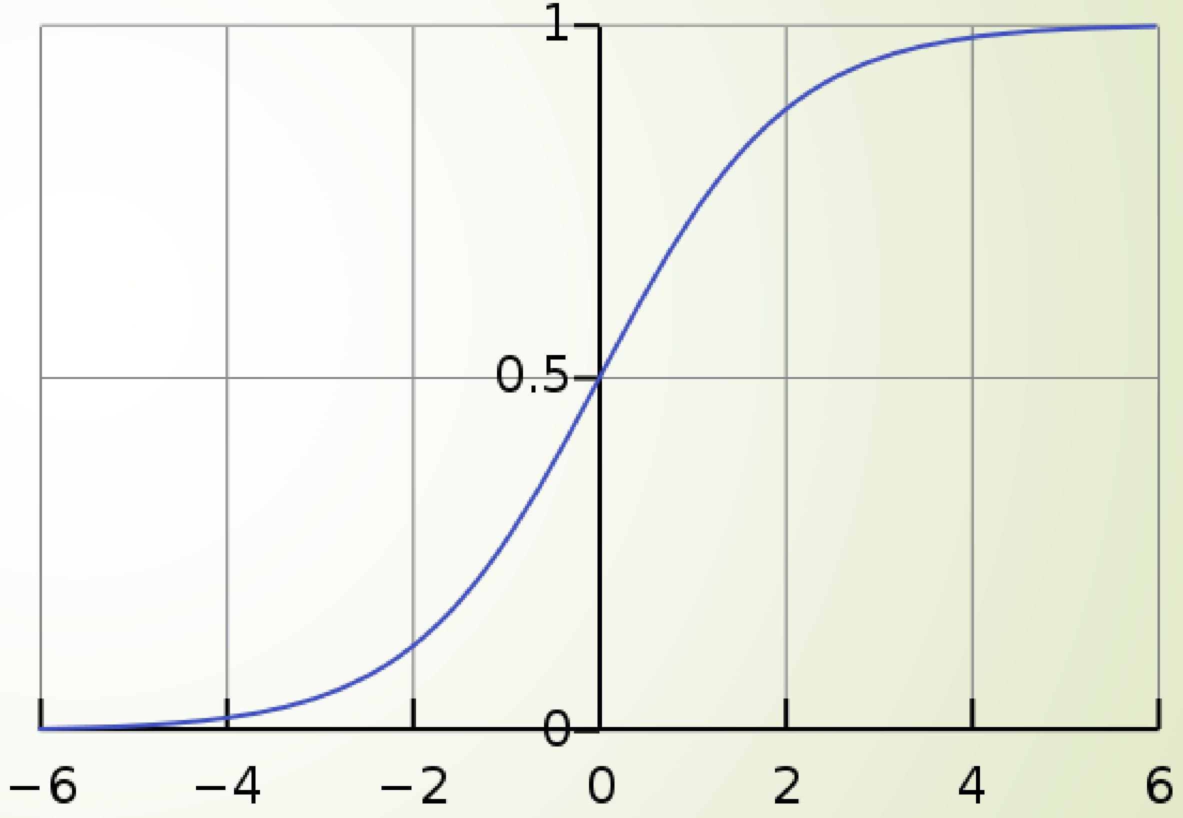
Household data

id	wBXbHZmp_DkQlr	wBXbHZmp_JhtDR	SlDKnCuu_GUusz	SlDKnCuu_alLXR	KAJOWiiw_BlZns	KAJOWiiw_TuovO	KAJOWiiw_rqUAG
14	0	1	1	0	0	1	0
18	0	1	1	0	0	1	0
36	0	1	1	0	1	0	0

Merge

Logistic Regression

- Sigmoid Function:
- $g(z) = \frac{1}{1+e^{-z}}$



- Derivative of Sigmoid Function:
- $g'(z) = g(z)\big(1 - g(z)\big)$

Logistic Regression

- *In logistic regression, we define:*

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

$$\begin{cases} P(y=1 \mid x\, ;\, \theta) = h_\theta(x) \\ P(y=0 \mid x\, ;\, \theta) = 1 - h_\theta(x) \end{cases}$$

$$\Longrightarrow \quad P(y \mid x\, ;\, \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$$

Logistic Regression

- *The likelihood of the parameters is,*

- $L(\theta)=P(\hat{y}\mid x\,;\,\theta)=\prod_{i=1}^{m} h_\theta\left(x^i\right)^{y(i)}\left(1-h_\theta\left(x^i\right)\right)^{1-y(i)}$

- *Maximize the log likelihood,*

- $\ell(\theta)=\log L(\theta)=\sum_{i=1}^{m} y^i \log h\left(x^i\right)+\left(1-y^i\right)\log\left(1-h\left(x^i\right)\right)$

Use gradient ascent to maximize log-likelihood

- *Calculate the partial derivative:*

- $\frac{\partial \ell(\theta)}{\partial \theta_j} = \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x)$

  $= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial \theta_j}\theta^T x$

  $= (y(1-g(\theta^T x)) - (1-y)\, g(\theta^T x))x_j$

  $= (y - h_\theta(x))\, x_j$

$$\Longrightarrow \text{Updating Rule: } \boxed{\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i))x_j^i}$$

Realize Logistic Regression in Python

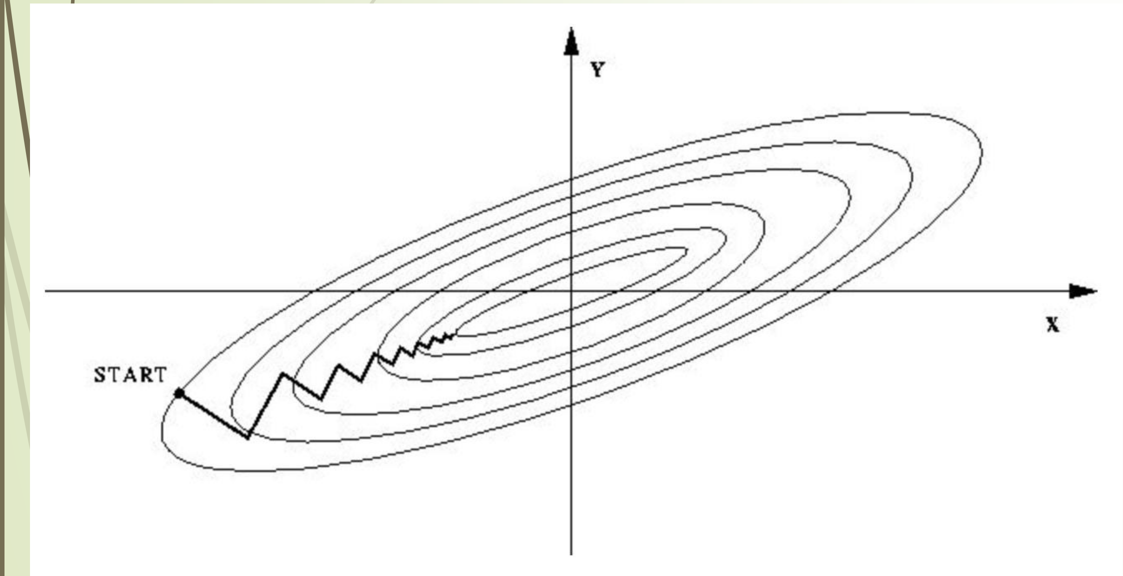
```python
def sigmoid(z):
    return 1/(1 + np.exp(-z))

def trans_data(data):
    data=data.as_matrix()
    features=data[:,0:-1]
    response=data[:,-1]
    features=np.insert(features,0,1,axis=1)
    return features,response

def gradient_descent(features,response,a,t):
    response = response.reshape(len(response),1)
    n=np.shape(features)[-1]
    #initialize weights
    weights = np.zeros((n, 1))
    #iteration times
    iterations = t
    acc=[]
    ll=[]
    for i in range(iterations):
        alpha=a
#        alpha=a/(1+i)+0.01
        #learning rate
        y = sigmoid(features.dot(weights))
        weights = weights + alpha * features.transpose().dot((response - y))

        y_pred_prob=sigmoid(features.dot(weights))
        ll.append(log_loss(response,y_pred_prob))
        y_pred=(y_pred_prob>0.5)*1
        acc.append(np.mean(y_pred==response))
#        acc.append(np.mean(y_pred==response))
    return weights,ll,acc
```

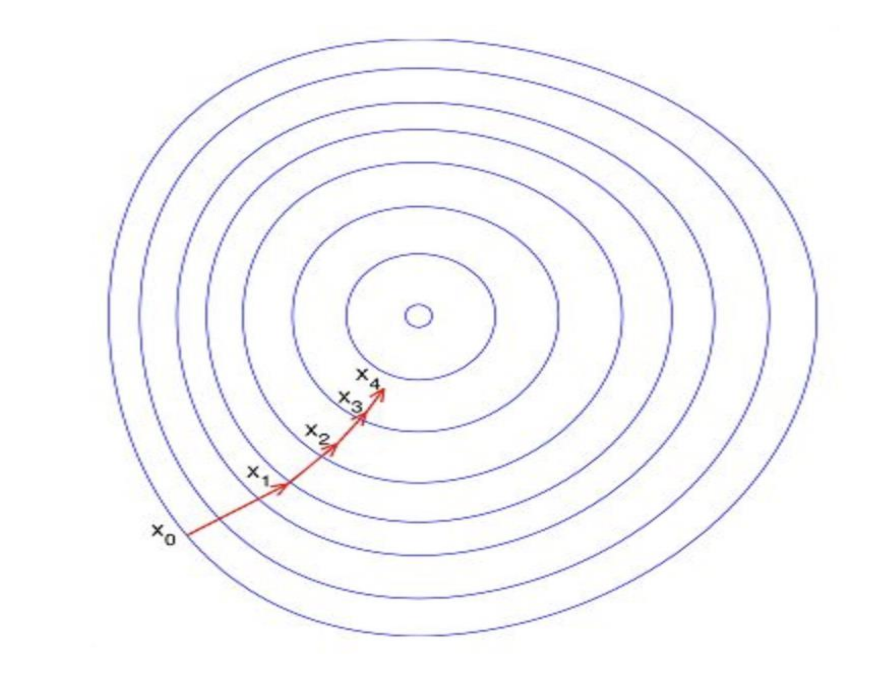
Vectorization

```python
def normalize(data):
    data=data.apply(lambda x:(x-x.mean())/x.std())
    return data

def LR_GD(data,a,t):
    data.iloc[:,:-1]=normalize(data.iloc[:,:-1])
    data=pd.concat([data.iloc[:,:-1],data.iloc[:,-1]],axis=1)
    features,response=trans_data(data)
    weights,ll,acc=gradient_descent(features,response,a,t)
    return weights,ll,acc

def LR_SGD(data,a,t):
    features,response=trans_data(data)
    weights,ll,acc=stochastic_gradient_descent(features,response,a,t)

    return weights,ll,acc

def visualize(data,a,t):
    weights=LR(data,a,t)
    f1=plt.figure(figsize=(8,8))
    s1=d1[d1.Status==1]
    s0=d1[d1.Status==0]
    plt.scatter(s1.iloc[:,0],s1.iloc[:,1],marker='o')
    plt.scatter(s0.iloc[:,0],s0.iloc[:,1],marker='x')
    plt.legend(loc='upper right')
    plt.title('Admission Status')
    plt.xlabel('Feature X1')
    plt.ylabel('Feature X2')
    x=np.linspace(1,100,100)
    y=-weights[1]/weights[2]*x-weights[0]/weights[2]
    plt.plot(x,y)
```

Feature scaling

Gradient Descent converges much faster with feature scaling than without it.





contour of the cost function: ‘oval shaped’

contour of the cost function: ‘circle shaped’

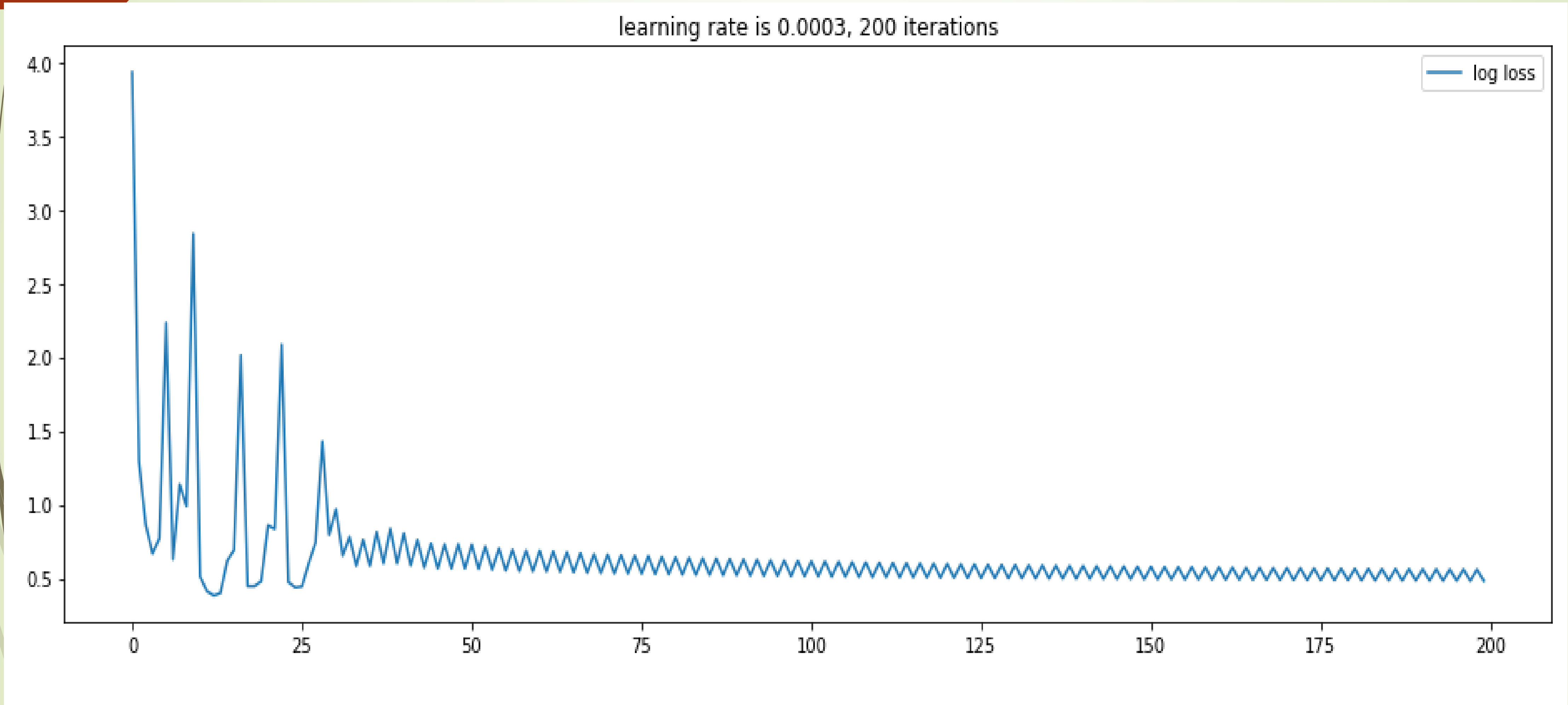
Feature scaling

- Before input the data into model, we need to standardize the data first.

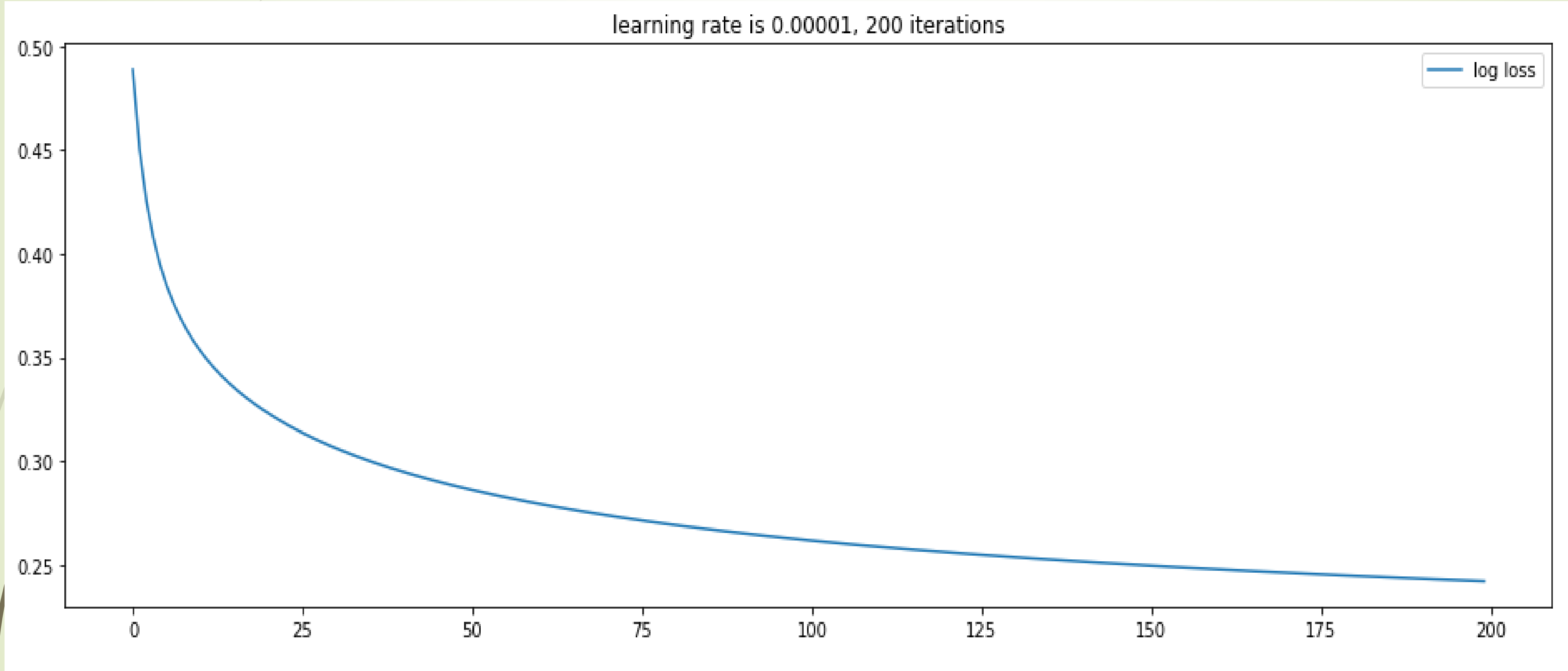
$$z_{ij} = \frac{x_{ij} - \overline{x_j}}{s_j}$$

- x_{ij} is j^{th} data point in feature i
- $\overline{x_j}$ *is the sample mean*
- s_j *is the standard deviation*

Experiments: 0.0003 learning rate, 200 iterations



Experiments: 0.0001 learning rate, 200 iterations



Results and comparison

	Our model	LogisticRegression in scikit-learn
Training Log Loss	0.1903	0.1901
Test Log Loss	0.3725	0.3687

Optimize the model by introducing Regularization

- Cost Function with L1 Regularization
- $J(\theta) = \underbrace{\boxed{-\frac{1}{m}\left[\sum_{i=1}^{m} y^i \log h\left(x^i\right) + (1-y^i)\log(1-h\left(x^i\right))\right]}}_{\text{Negative log-likelihood}} + \underbrace{\boxed{\frac{\lambda}{m}\sum_{j=1}^{n} |\theta_j|}}_{\text{Regularization term}}$

- Cost Function with L2 Regularization
- $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^i \log h\left(x^i\right) + (1-y^i)\log(1-h\left(x^i\right))\right] + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2}$

Minimize the cost function using Newton's method

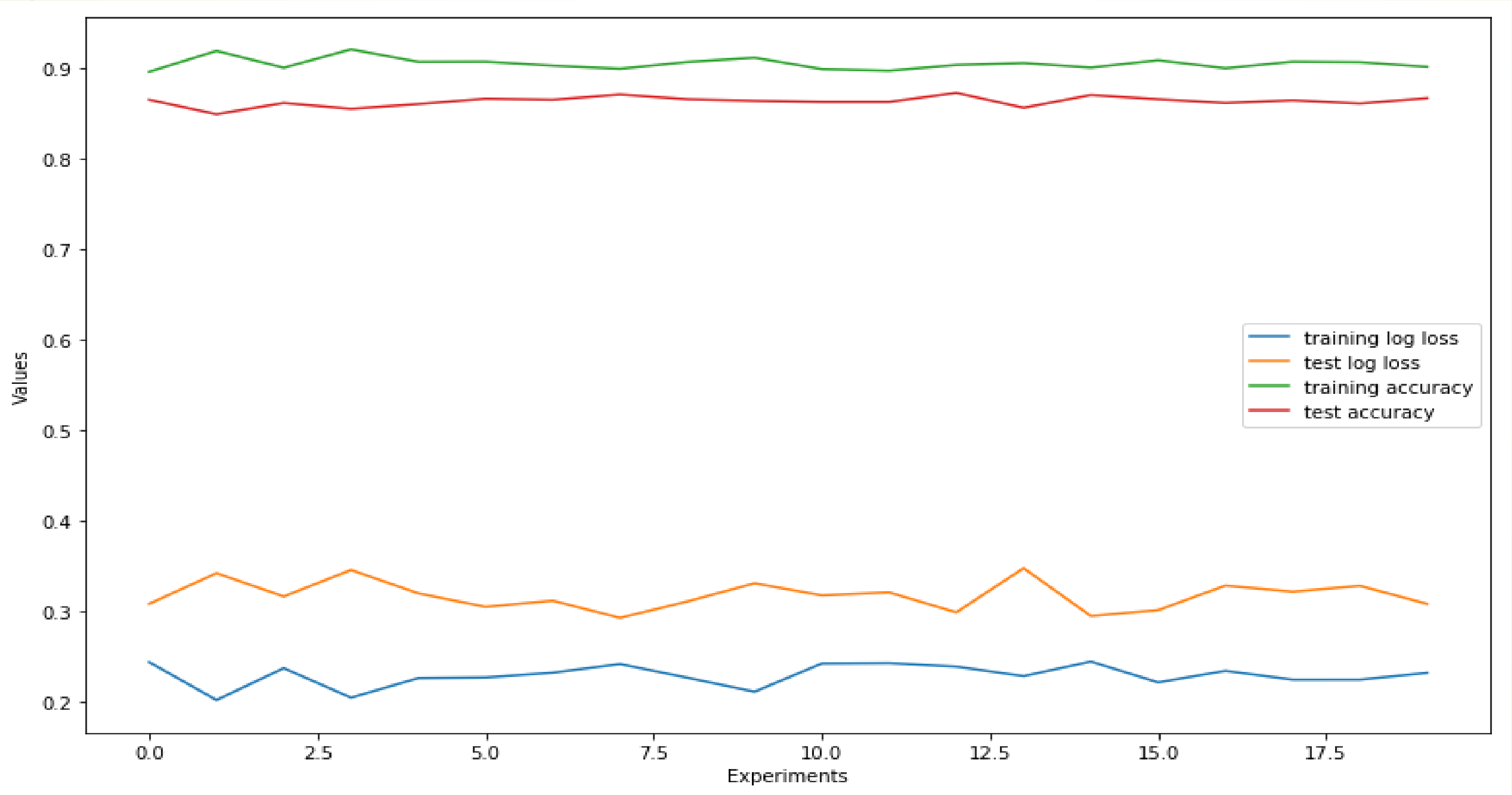
- $$\nabla_\theta J = \begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right){x_0}^{(i)} \\ \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right){x_1}^{(i)} + \frac{\lambda}{m}\theta_1 \\ \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right){x_2}^{(i)} + \frac{\lambda}{m}\theta_2 \\ \vdots \\ \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right){x_n}^{(i)} + \frac{\lambda}{m}\theta_n \end{bmatrix}$$ Gradient

- $$H = \frac{1}{m}\left[\sum_{i=1}^{m} h_\theta\left(x^{(i)}\right)\left(1 - h_\theta\left(x^{(i)}\right)\right)x^{(i)}\left(x^{(i)}\right)^T\right] + \frac{\lambda}{m}\begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 1 \end{bmatrix}$$ Hessian matrix

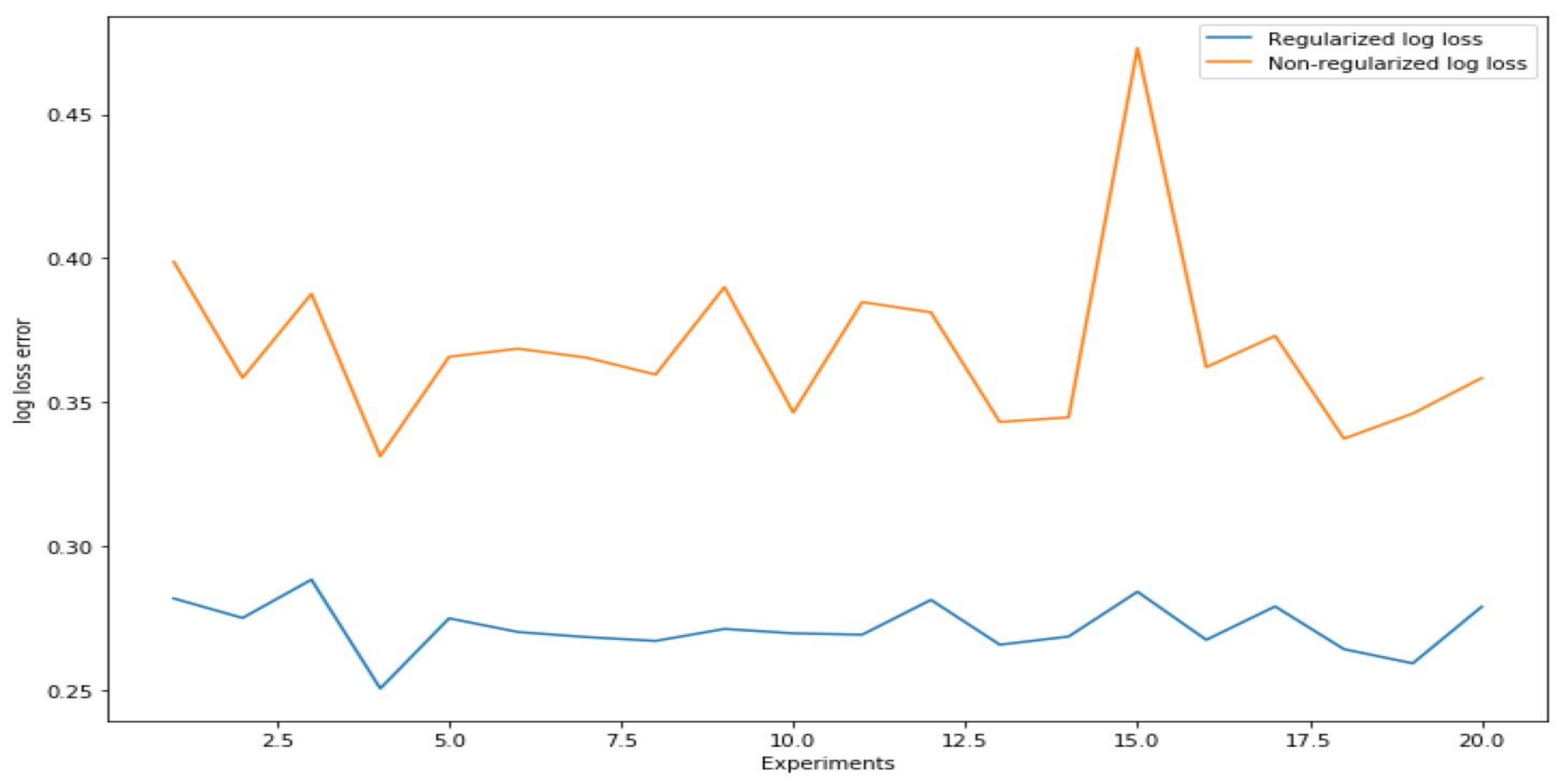
- *Updating Rule:* $\boxed{\theta^{(t+1)} = \theta^{t} - H^{-1}\nabla_\theta J}$

Results of model with regularization

Metrics (in average)	Training	Test
Log Loss	0.2294	0.2684
Accuracy	90.55%	86.38%



Comparison of regularized and non-regularized model



From the experiment we can see that the regularized model outperforms the non-regularized logistic regression.