Shiraz University

گزارش تمرین دوم

(شناسایی آماری الگو)

تهیه شده توسط:

سارا لیمویی، ملیکا زارع

استاد:

دکتر عظیمی فر

پاییز 1400


بسم الله الرحمن الرحیم

فهرست مطالب

پیش گفتار

در این تمرین قصد داریم به کمک logistic Regression، classification انجام دهیم. برای این منظور، ابتدا با Binary Classification شروع می کنیم.

لازم به ذکر است که در همه ی قسمت ها از دیتاست Iris استفاده می کنیم.

آماده سازی داده ها

مجموعه ی داده ی Iris، یک مجموعه ی 150 تایی از 3 کلاس مختلف می باشد. ابتدا آن را به کمک pandas از لینک ذکر شده در صورت تمرین، مجموعه دیتاست های UCI، بارگذاری می کنیم. از آنجا که تمرین قسمت های متفاوتی دارد، مجموعه ی داده ای که برای هر قسمت استفاده می شود متفاوت خواهد بود که در ادامه در هر بخش، آن را توضیح داده ایم.

Binary Classification

در این قسمت، می خواهیم یک binary classifier را پیاده سازی کنیم. همان طور که از اسم آن مشخص است، در اینجا به دو کلاس داده نیاز داریم. پس باید رکورد های مربوط به یکی از کلاس های مجموعه داده ی Iris را حذف کنیم که در این تمرین، کلاس "Iris-versicolor" را حذف می کنیم. لازم به ذکر است که تمامی تابع های آماده سازی داده ها، در فایل PreprocessData و کلاس IrisDataset پیاده-سازی شده اند. همچنین از آنجا که دیتاست Iris، دارای 4 ستون می باشد، ستون های سوم و چهارم آن را نیز حذف می کنیم.

مشابه با تمرین قبل، میتوان نوع normalization داده ها را نیز مشخص کرد و یا هیچ نرمال سازی روی داده ها اعمال نکرد. همچنین 80٪ داده ها را برای train و 20٪ را برای test جدا می کنیم.

اکنون که داده ی ما آماده است، سراغ binary classifier می رویم.

تابع h_theta():

در این تمرین، تابع sigmoid به عنوان hypothesis-function در نظر گرفته شده است. در نتیجه برای پیدا کردن احتمال آنکه X از کلاس 1 باشد، h_theta(X) را صدا میزنیم که به صورت زیر تعریف می شود:

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

g() در اینجا همان sigmoid می باشد که هنگام تست binary classifier و ساختن یک object از کلاس آن، به عنوان پارامتر به constructor کلاس داده می شود.

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

iris_dataset = IrisDataset(change_to_binary_classed_data=True, normalization_method='scale_0_1')

binary_classifier = BinaryClassifier(hypothesis_fn=sigmoid, cost_fn='cross_entropy', alpha=1e-2, max_iter=30000)
```

تابع likelihood():

در مساله های با دو کلاس، از $P(y|X; \theta)$ استفاده می کنیم و نهایتا از روش *gradient ascent* برای ماکسیمایز کردن *likelihood* استفاده می کنیم. *likelihood* به صورت زیر تعریف می شود:

$$P(y|X;\theta) = h_\theta(X)^y(1 - h_\theta(X)^{1-y})$$

$$L(\theta) = \prod_{j=1}^{m} P(y^{(j)}|X^{(j)};\theta)$$

$$= \prod_{j=1}^{m} h_\theta(X^{(j)})^{y^{(j)}}(1 - h_\theta(X^{(j)})^{1-y^{(j)}}))$$

برای راحتی کار در مشتق گیری، به جای خود likelihood، از Log(likelihood) یعنی l استفاده می کنیم که به صورت زیر تعریف می شود:

$$l(\theta) = \log(L(\theta)) = \sum_{j=1}^{m} y^j log(h_\theta(X^j)) + (1 - y^j) log(1 - h_\theta(X^j))$$

$l(\theta)$ در هر مرحله در یک آرایه قرار میگیرد و نهایتا مشاهده می کنیم که با گذشت هر iteration و update کردن مقادیر theta، مقدار likelihood ماکسیمایز می شود.

تابع gradient_of_likelihood ():

در این تابع، مشتق likelihood به ازای هر یک از پارامتر های مدل محاسبه می شود و نهایتا به کمک فرمول زیر، در محاسبه و update کردن مقادیر theta، مورد استفاده قرار می گیرد.

$$\theta_i = \theta_i + \alpha \sum_{j=1}^{m} (y^{(j)} - h_\theta(X^{(j)})) x_i^{(j)}$$

تابع هزینه:

در این تمرین از cross-entropy به جای تابع های error دیگر از جمله mse استفاده کردیم. mse یا همان mean square error، مربوط به فضای پیوسته می باشد در صورتی که در اینجا در فضای logistic و گسسته هستیم. در واقع با استفاده از mse در مسائل logistic، به مقدار error صحیحی نمی-رسیم چرا که بر اساس فرمول mse که در زیر آمده است، با در نظر گرفتن y_hat=1 و y=0، تفاوت آنها یک واحد خواهد بود که ممکن است در واقعیت مقدار error به این معنا و نسبت نباشد.

$$mse: J(\theta) = \frac{1}{m} \sum_{j=1}^{m} \left(\hat{y}^j - y^j\right)^2$$

با محاسبه ی ارور از طریق فرمول cross entropy، میتوان مقدار دقیق تری برای ارور پیدا کرد و مطمئن میشویم که مدل همیشه converge کند.

$$\text{cross entropy} = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}$$

تابع fit():

در این تابع، مدل ما train می شود. ابتدا theta ی اولیه را با صفر مقدار دهی می کنیم. سپس با حلقه الگوریتم gradient ascent را پیاده کرده ایم.

تابع predict():

در این تابع، بر اساس احتمالی که برای هر یک از داده های X، که از کلاس 1 باشند یا از کلاس 0، label نهایی آن ها یعنی class-0 و class-1 را پیدا می کنیم. به این صورت عمل می کنیم که اگر احتمال از کلاس 1 بودن، بیشتر از 0.5 بود، کلاس داده ی مذکور را 1 و اگر کوچکتر یا مساوی 0.5 بود، کلاس آن را 0 در نظر می گیریم.

نهایتا یک آرایه به ازای همه ی داده های X، به عنوان خروجی به ما می دهد.

تابع report() و plot():

مشابه تمرین قبل، در این دو تابع پارامتر های اصلی مدل و مقدار cross entropy بر روی هر یک از داده های تست و train گزارش می شوند. همچنین در این تمرین، معادله ی decision boundary نیز نمایش داده می شود.

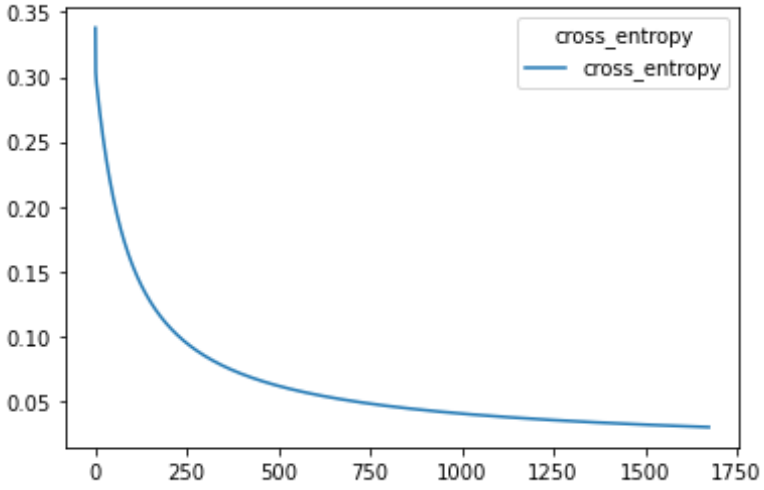
در تابع plot نیز، با استفاده از کتابخانه ی matplotlib، داده های هر کلاس با رنگ های متفاوت و decision boundary نیز به صورت نمودار plot می شوند.

نتایج Binary Classification

مقدار likelihood به ازای هر 500 iteration:

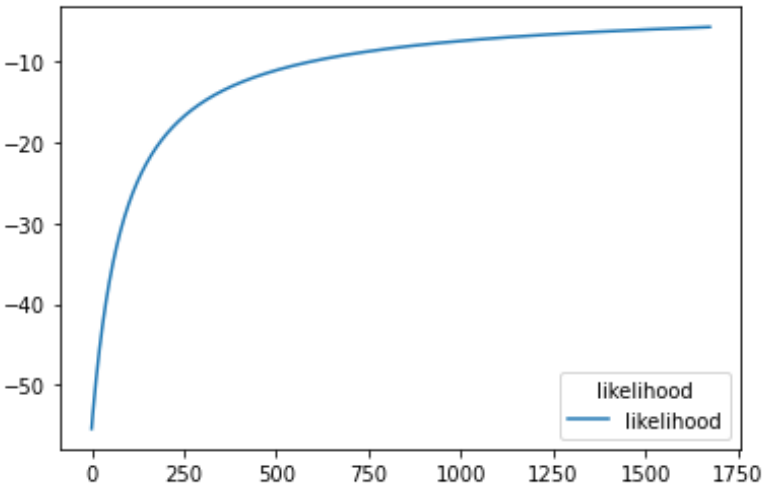
```
No normalization.
likelihood in iteration 0: -55.451774444795625
likelihood in iteration 500: -11.043631967919154
likelihood in iteration 1000: -7.417016239048268
likelihood in iteration 1500: -6.006371297986199
```

نمودار cross entropy:

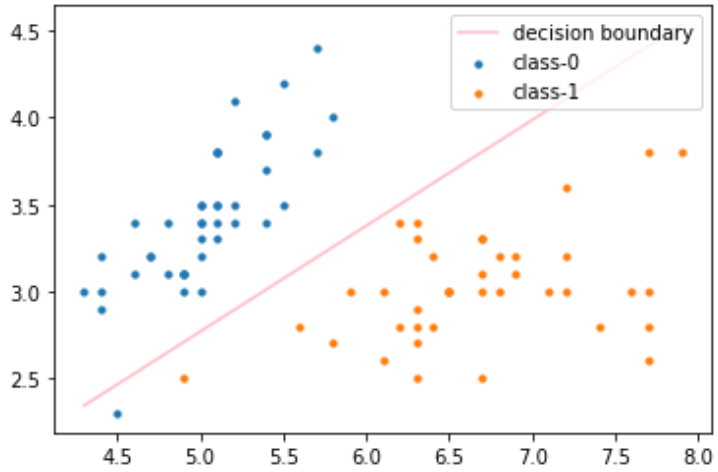



```
Iteration 1674 with difference 9.993849184136011e-06 converged.
```

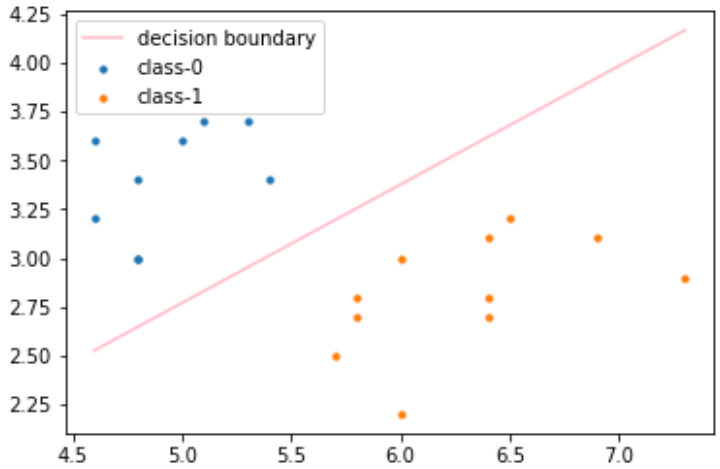
نمودار likelihood:



نمودار داده های train:



نمودار داده های test:



Report:

```
theta_ 0 :  -1.4627012203421768
theta_ 1 :  3.2692857359692886
theta_ 2 :  -5.375545751538667
Cross-Entropy on data train:  0.030430039426007128
Cross-Entropy on data test:  0.022272507851832973
Accuracy metric on data train:  98.75
Accuracy metric on data test:  100.0
Decision Boundary Euqation: y = 0.61*x + -0.27
```

Multi-class Classification

در این قسمت، می خواهیم یک multi-class classifier را پیاده سازی کنیم. در این جا می خواهیم به کمک binary classification و دو روش one vs. one و one vs. all، این classifier را پیاده سازی کنیم.

در این جا، نیازی به حذف feature های سوم و چهارم و یا بعضی از sample ها نداریم. بلکه از کل دیتاست به نسبت 80٪ داده ی train و 20٪ داده ی تست، استفاده می کنیم.

کلاس MulticlassClassificationLogisticRegression جهت این قسمت پیاده سازی شده است. با انتخاب پارامتر method هنگام ساختن object از این کلاس، روش را مشخص می کنیم (one vs. one و یا one vs. all)

روش one vs. one:

در این روش، به تعداد num_of_class * (num_of_class – 1) / 2 دیتاست و binary classifier تعریف می کنیم. سپس مدل را به ازای هر جفت کلاس های دیتاست، آموزش می دهیم. برای پیدا کردن label مربوط به یک x، داده ی x را به عنوان ورودی به predict هر یک از مدل ها می دهیم و با توجه به کلاسی که به عنوان label بر میگرداند، یه واحد به تعداد predict شده های آن کلاس اضافه می شود. در نهایت آن کلاسی که بیشترین تعداد بار predict شده، به عنوان label داده ی x در نظر می گیریم.

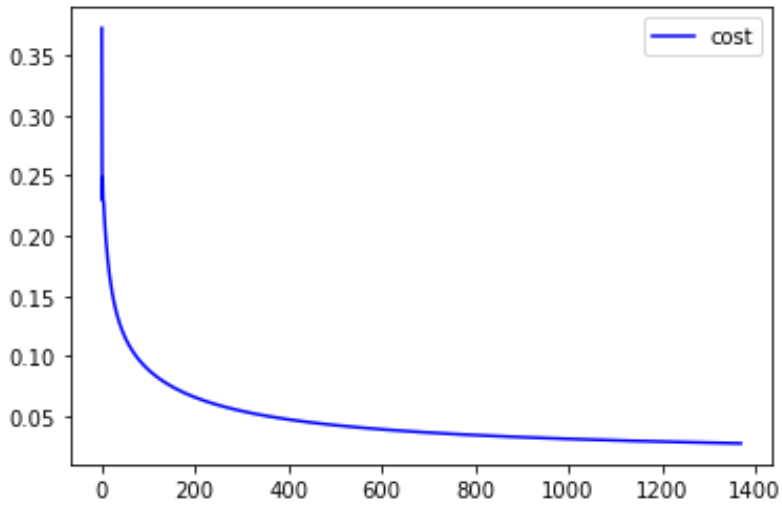
نتایج روش one vs. one:

Multi-class Classifier, 'One vs. One'

```
iris_dataset = IrisDataset(change_to_binary_classed_data='multiclass', normalization_method='none')

multinomial_classifier = MulticlassClassificationLogisticRegression()
multinomial_classifier.fit(method='one_vs_one', normalization_method='none', cost_fn='cross_entropy',
                                    alpha=1e-3, max_iter=50000, eps=1e-6)

multinomial_classifier.predict_one_vs_one(iris_dataset.x_train)
multinomial_classifier.predict_one_vs_one(iris_dataset.x_test)

multinomial_classifier.report(iris_dataset.x_train, iris_dataset.y_train, iris_dataset.x_test, iris_dataset.y_test)

multinomial_classifier.plot_cost()
```

```
No normalization.
No normalization.
No normalization.
No normalization.
Creating binary classifiers...
Training binary classifiers...
Iteration 8902 with difference 9.999464529730773e-07 converged.
Iteration 1369 with difference 9.991120205939813e-07 converged.
Iteration 2106 with difference 9.99535131006573e-07 converged.
Mean Convergence iteration  4125.666666666667
Accuracy on data train:  97.5
Accuracy on data test:  96.66666666666667
```

نمودار هزینه:



روش one vs. all یا one vs. rest:

رویکرد ovr مجموعه داده را به یک کلاس در مقابل 2 کلاس دیگر تقسیم می کند. بنابراین تعداد مجموعه داده هایی که باید در این روش ایجاد شوند به تعداد تعداد کلاس ها می شود. در این سوال، از آنجایی که ما با iris-dataset کار می کنیم، دارای 3 کلاس یا برچسب است. بنابراین، تعداد مجموعه داده های ایجاد شده 3 است .پس از تهیه این 3 مجموعه داده، 3 مدل طبقه بندی باینری را با هر مجموعه داده آموزش می دهیم. برای پیش بینی کلاس Xj ، احتمال وجود از هر کلاس را در مقابل دو کلاس دیگر با استفاده از کلاسفایر

باینری پیش بینی می کنیم.سپس کلاسی را که حداکثر احتمال را به عنوان کلاس Xj دارد حدس نهایی مدل ovr گزارش می دهیم.

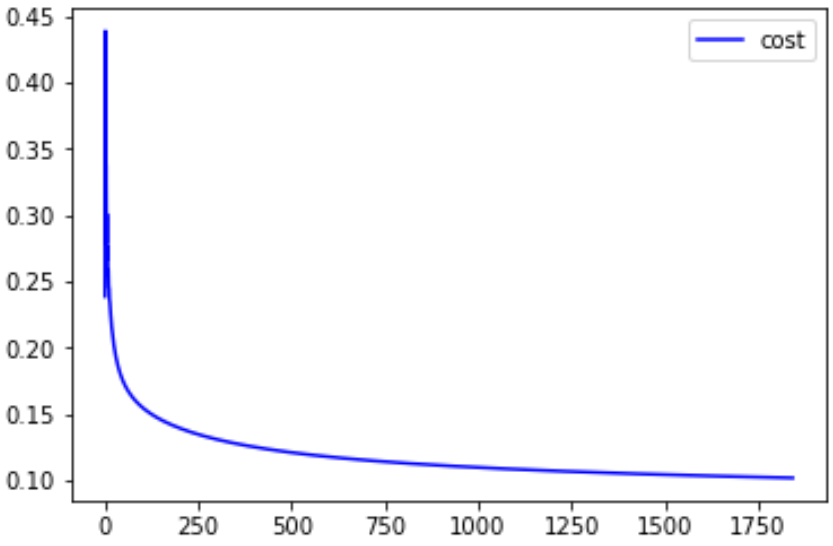
نتایج روش one vs. all:

Multi-class Classifier, 'One vs. All'

```
iris_dataset = IrisDataset(change_to_binary_classed_data='multiclass', normalization_method='none')

multinomial_classifier = MulticlassClassificationLogisticRegression()
multinomial_classifier.fit(method='one_vs_all' , normalization_method='none', cost_fn="cross_entropy",
                                    alpha=1e-3, max_iter=50000, eps=1e-6)

multinomial_classifier.predict_one_vs_all(iris_dataset.x_train)
multinomial_classifier.predict_one_vs_all(iris_dataset.x_test)

multinomial_classifier.report(iris_dataset.x_train, iris_dataset.y_train, iris_dataset.x_test, iris_dataset.y_test)

multinomial_classifier.plot_cost()
```

```
No normalization.
No normalization.
No normalization.
No normalization.
Creating binary classifiers...
Training binary classifiers...
Iteration 1843 with difference 9.998441777552686e-07 converged.
Iteration 7795 with difference 9.997727984378812e-07 converged.
Iteration 7942 with difference 9.998425389810828e-07 converged.
Mean Convergence iteration  5860.0
Accuracy on data train:  98.33333333333333
Accuracy on data test:  96.66666666666667
```

نمودار هزینه:



طبقه بندی چندکلاسه با استفاده از softmax

از آنجایی که ما به طور کلی k کلاس داریم، پارامتر مدنظر ما از توزیع چندجمله‌ای پیروی می‌کند. برای تصمیم گیری در مورد پارامترها، می توانیم k پارامترهای ϕ_1، ϕ_2،...، ϕ_kرا انتخاب کنیم. برای مدل چندجمله ای ما داریم:

$$P(y=1|X;\theta)=\phi_1$$

$$P(y=2|X;\theta)=\phi_2$$

$$\vdots$$

$$P(y=k|X;\theta)=\phi_k$$

که مجموع ϕ_iها یک می شود.

با جایگزینی آن در generalized linear model داریم:

$$\phi_i = \frac{e^{\eta_i}}{\sum_{l=1}^{k} e^{\eta_l}} (i = 1, \cdots, k)$$

با فرض $\forall i=1,2,...,k$ مدل رگرسیون softmax به صورت زیر در می آید:

$$P(y = i|x; \theta) = \frac{e^{\theta_i^T x}}{\sum_{l=1}^{k} e^{\theta_l^T x}} = \phi_i$$

برای یافتن مقادیر بهینه θ ، با تکنیک پیدا کردن حداکثر مقدار تابع likelihood ، θایی مدنظر بدست می-آید. برای یافتن ماکزیمم این تابع، ما با تابع log (likelihood) شروع می کنیم.

$$\ell(\theta) = \sum_{i=1}^{m} \log P\left(y^{(i)}|x^{(i)};\theta\right)$$

$$= \sum_{i=1}^{m} \log \prod_{j=1}^{k} \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \right)^{1\{y^{(i)}=j\}}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{k} \log \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \right)^{1\{y^{(i)}=j\}} = \sum_{i=1}^{m} \sum_{j=1}^{k} \log(\phi_j)^{1\{y^{(i)}=j\}}$$

برای پیدا کردن مشتق تابع بالا ابتدا مشتق تابع softmax را با توجه به ϕ پیدا می کنیم:

$$\frac{\partial}{\partial \theta_p}(\phi_j) = \frac{\partial}{\partial \theta_p}\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right) = \phi_j(1\{p = j\} - \phi_p)x^{(i)}$$

حال دوباره به تابع log (likelihood) برمی گردیم. در ادامه مشتق آن را تمام می کنیم:

$$\frac{\partial}{\partial \theta_p}\left(\sum_{j=1}^{k} \log(\phi_j)^{1\{y^{(i)}=j\}}\right) = \left(1\{y^{(i)} = p\} - \phi_p\right) x^{(i)}$$

$$\frac{\partial}{\partial \theta_p}\ell(\theta) = \frac{\partial}{\partial \theta_p}\left(\sum_{i=1}^{m}\sum_{j=1}^{k} \log(\phi_j)^{1\{y^{(i)}=j\}}\right) = \sum_{i=1}^{m}(1\{y^{(i)} = p\} - \phi_p)x^{(i)}$$

در نهایت کافی است روش gradient ascent را پیاده سازی کنیم تا مقدار بهینه θ را بدست آوریم.

$$\theta_p = \theta_p + \alpha \sum_{i=1}^{m}(1\{y^{(i)} = p\} - \phi_p)x^{(i)}$$

کلاس SoftmaxLogisticRegression

در این کلاس توابع مورد نیاز برای پیاده سازی مدل softmax قرار گرفته اند. در تابع fit با گرفتن داده ی آموزشی عملیات gradient ascent را انجام داده ایم. در هر حلقه از این الگوریتم مقدار θ را به روز می کنیم. هم چنین با استفاده از تابع هزینه مشخص شده بین احتمال کلاس حدس زده شده و لیبل کلاس واقعی، هزینه را حساب کرده و ذخیره می کنیم. در نهایت اگر حلقه فعلی با قبلی نتایج بهتری ارائه ندهد و تفاوت هزینه با حلقه قبل از مقدار اپسیلون مشخص شده کمتر شود، الگوریتم converged کرده و مقدار بهینه θ پیدا شده است. پس الگوریتم را متوقف می کنیم.

در تابع phi نیز مدل احتمال softmax را پیاده کرده ایم. با حساب کردن $\theta^T x$ و دادن به تابع softmax احتمال لیبل هر کلاس برای هر داده را بدست می آوریم.

در تابع predict نیز با صدا زدن تابع phi احتمال هر کلاس را گرفته و با عمل argmax کلاسی که بیشترین احتمال دارد را مشخص می کنیم.

در تابع accuracy_metric نیز لیبل های حدس زده شده مدل را با مقدار واقعی آنها مقایسه کرده و درصد درستی آن را گزارش می دهیم.

در تابع cross_entropy_cost نیز تابع هزینه پیاده شده است. دلیل استفاده از این تابع هزینه نیز در بالا گفته شده است.

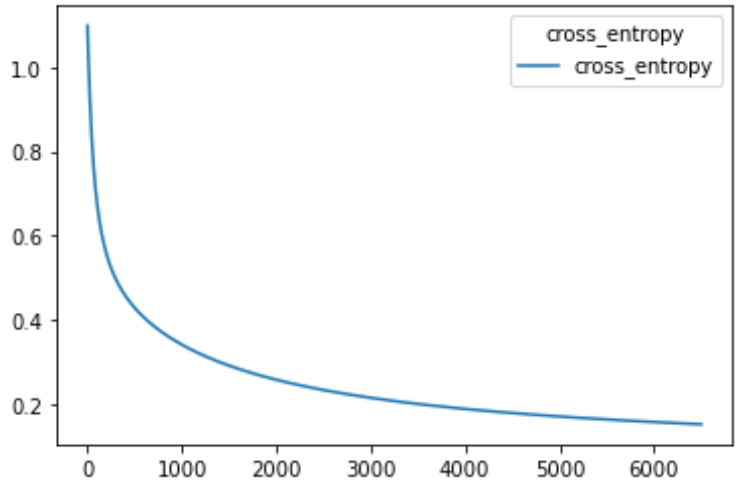
در نهایت با فراخوانی تابع report می توان تمامی اطلاعات مدل آموزش داده شده را مشاهده کرد. هم چنین مقدارهای هزینه بر روی داده آموزشی و تست و درصد درستی بر روی داده آموزشی و تست را گزارش داده ایم. مقدار بهینه θ برای هر کلاس را نیز چاپ کرده ایم.

نتایج طبقه بندی با softmax

```
iris_dataset = IrisDataset(change_to_binary_classed_data='multiclass', normalization_method='none')

softmax_classifier = SoftmaxLogisticRegression()
softmax_classifier.fit(iris_dataset.x_train, iris_dataset.y_train,iris_dataset.x_test, iris_dataset.y_test,
                   n_iter= 20000, alpha = 1e-4, eps=1e-5, cost_func='cross_entropy')

softmax_classifier.predict(iris_dataset.x_test)

softmax_classifier.report(iris_dataset.x_train, iris_dataset.y_train,iris_dataset.x_test, iris_dataset.y_test)
softmax_classifier.plot_cost()
```

```
No normalization.
Training softmax classifier...
Iteration 6497 with difference 9.999599712345875e-06 converged.


Accuracy metric on data train:  97.5
Accuracy metric on data test:  100.0
Cross-Entropy on data train:  9.812276905510183
Cross-Entropy on data test:  12.43948352318164



theta for class  0 :
    theta_ 0 :  0.3758730864171454
    theta_ 1 :  0.8063604510229611
    theta_ 2 :  1.8827241517351478
    theta_ 3 :  -2.5908530606913414
    theta_ 4 :  -1.2221486891752826
theta for class  1 :
    theta_ 0 :  0.5350060193871613
    theta_ 1 :  0.6485065736200177
    theta_ 2 :  -0.23854515960507733
    theta_ 3 :  -0.06894189797190965
    theta_ 4 :  -0.94831423908457
theta for class  2 :
    theta_ 0 :  -0.9108791058043022
    theta_ 1 :  -1.4548670246429771
    theta_ 2 :  -1.644178992130067
    theta_ 3 :  2.659794958663246
    theta_ 4 :  2.1704629282598478
```

در نهایت با فراخوانی تابع plot_cost نمودار تغییرات هزینه در هر دوره (iteration) را داریم.



مقایسه:

نتایج بدست آمده سه روش طبقه بندی چند کلاسه را در جدول زیر مشاهده می کنید الگوریتم Regression Softmax کمی بهتر از دیگر روش ها در داده های Test عمل کرده است اما در داده های Train هر سه روش accuracy تقریبا یکسانی دارند.

Test	Train	Accuracy
96.66	97.5	One vs. one
96.66	98.33	One vs. all
100	97.5	Softmax