گزارش پیاده‌سازی الگوریتم‌های
Logistic Regression
و
Softmax

محمد شعاعی امیر محمد مزارعی

۱۶ آذر ۱۴۰۰

بخش اول
توابع و کلاس‌ها

برای هر کدام از قسمت‌های مورد نیاز یک کلاس جدا پیاده سازی شده است. کلاس ها و متدهای مربوط به هر کلاس به شرح زیر است:

۱. `BinaryClassifier(eta, tol, n_iters)` : این کلاس پیاده سازی classification دو کلاسه است که شامل متدهای زیر است:

- `fit(X, y)`
- `predict(X)`
- `predict_proba(X)`
- `cost(X, y)`
- `sigmoid(z)`

۲. `MulticlassClassifier(eta, tol, n_iters, solver)` : این کلاس شامل پیاده سازی دو روش OvO[۱] و OvR[۲] است که با ورودی solver کنترل می‌شود. همچنین از مقدار tol فقط در روش OvR استفاده می‌شود. این کلاس شامل متدهای زیر است:

- `fit(X, y)`

---
[۱] One-vs-One
[۲] One-vs-Rest

- `predict(X)`
- `predict_proba(X)`
- `cost(X, y)`
- `sigmoid(z)`

۳. `Softmax(eta, tol, n_iters)` : این کلاس شامل پیاده سازی روش softmax است. متدهای این کلاس به شرح زیر است:

- `fit(X, y)`
- `predict(X)`
- `predict_proba(X)`
- `cost(X, y)`
- `softmax(z)`

۴. سایر توابع کمکی که در قسمت‌های مختلف استفاده شده‌اند به صورت زیر هستند:

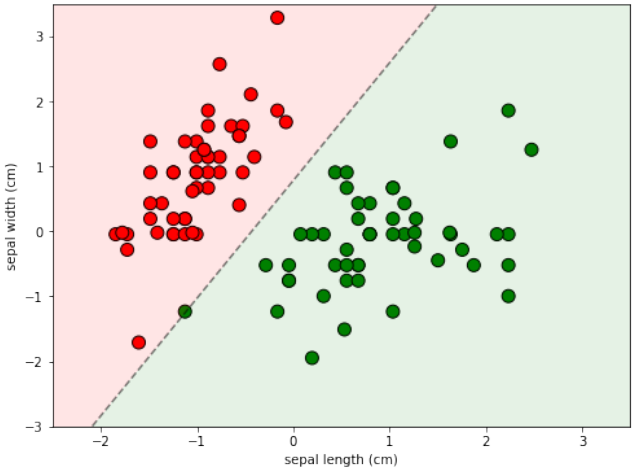
- `zero_mean_normalize(df, column)` : نرمال سازی مقدار ستون‌های مشخص شده در داده ورودی
- `cross_entropy_error(y_true, y_pred)` : محاسبه مقدار خطا برای logistic regression
- `one_hot_encoding(y_true, n_classes)` : تبدیل برچسب‌ها به شکل مناسب برای الگوریتم softmax
- `measure_accuracy(y_true, y_pred)` : محاسبه دقت مدل

بخش دوم

Regression Logistic

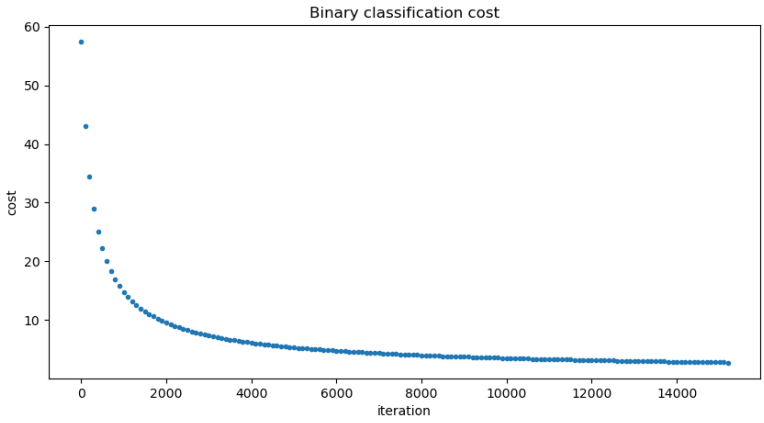
۱ Binary Classification

پس از اتمام آموزش مدل شکل ۱.۱ حاصل پیشبینی داده‌های train و test است.



شکل ۱.۱

نمودار مربوط به خطا که در هر تکرار محاسبه شده است در شکل ۲.۱ نمایش داده شده است.



شکل ۲.۱

خطای مدل بر روی داده‌های train و test در کد ۱ چاپ شده‌اند.

```
cross-entropy (train): 0.02060000069254901
cross-entropy (test): 0.00011377187125823006
theta: [[ 1.84193838]
        [ 4.25519669]
        [-2.35164349]]
```

کد ۱: خطاهای مربوط به مدل دو کلاسه

همچنین فرمول خط decision boundary به صورت معادله ۱ است:

$$y = 1.8095x + 0.7833 \quad (۱)$$

۲ Multiclass Classification

۱.۲ One-vs-Rest

این روش به تعداد کلاس‌های موجود در داده یک مدل دو کلاسه آموزش می‌دهد و یکی از مشکلات این روش نا متوازن شدن داده‌ها هنگام یادگیری است.
پس از آموزش مدل با ورودی‌های نمایش داده شده در کد ۲ مقادیر مربوط به میزان دقت هم چنین شماره‌ی تکرار همگرایی در کد ۳ چاپ شده‌اند.

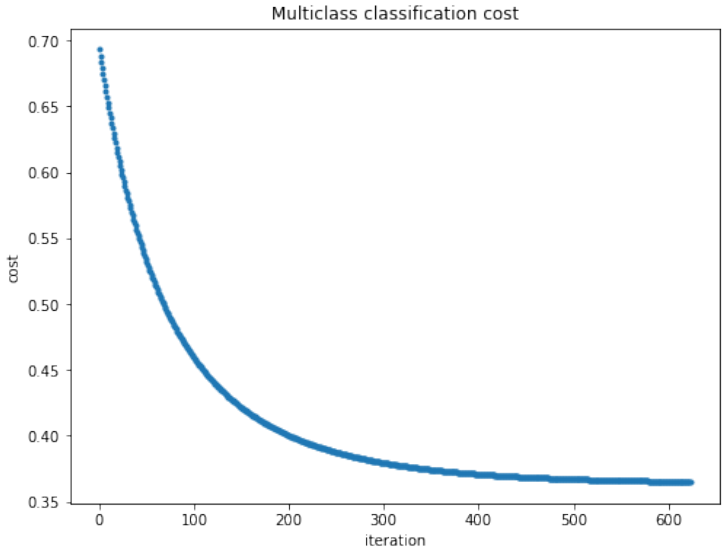
```
ovr = MulticlassClassifier(eta=1e-4, tol=1e-5, n_iters=1e4,
↪  solver='ovr')
ovr.fit(df_train.iloc[:, :-1].to_numpy(), df_train.iloc[:, -1])
```

کد ۲: ورودی‌های مدل OvR

```
accuracy (train): 0.8416666666666667
accuracy (test): 0.7666666666666667
convergence iteration: 623
theta:
[[-0.70711877 -0.63176098  0.74534982 -0.93384678 -0.87968063]
[-0.70249057   0.06307685 -0.7776572   0.11260695 -0.10848047]
[-0.64453415   0.48343572  0.09691334  0.76237581  0.9371994 ]]
```

کد ۳: میزان دقت مدل OvR و شماره تکرار همگرایی

نمودار مربوط به تابع هزینه در شکل ۱.۲ نمایش داده شده است.



شکل ۱.۲

One-vs-One ۲.۲

در این روش به ازای هر جفت کلاس یک مدل دو کلاسه آموزش داده می‌شود و بر خلاف روش OvR از نامتوازن بودن داده‌ها رنج نمی‌برد اما با افزایش تعداد کلاس‌ها زمان یادگیری به صورت نمایی افزایش خواهد یافت.
ورودی های مربوط به این مدل در کد ۴ آمده است. همچنین مقادیر مربوط به دقت مدل در کد ۵ نمایش داده شده است.

```
ovo = MulticlassClassifier(
        eta=1e-4, n_iters=1e5, solver='ovo')
ovo.fit(df_train.iloc[:, :-1].to_numpy(), df_train.iloc[:, -1])
```

کد ۴: ورودی‌های مدل OvO

```
accuracy (train): 0.975
accuracy (test): 0.8333333333333334
theta:
[[-3.85686431 -1.79558686  2.53491411 -3.56922839 -2.95230439]
[-1.37635674  -1.3988      1.41402754 -2.77723175 -2.92909212]
[ 7.48237415   0.86189624  0.97550359 -6.46540629 -6.8184238 ]]
```

کد ۵: میزان دقت مدل OvO

۳ Softmax

برای این الگوریتم تنها مقدار خواسته شده پس از آموزش مدل میزان دقت آن است که در کد ۶ چاپ شده است.

```
accuracy (train): 0.9833333333333333
accuracy (test): 0.9333333333333333
theta:
[[-0.67669442, -3.01944273,  2.64580279, -5.81525285,
↪  -5.26884148],
[ 6.42054669,  2.0871986 , -0.48711804, -2.5175024 ,
↪  -2.19956475],
[-5.74385227,  0.93224413, -2.15868475,  8.33275525,
↪  7.46840622]]
```

کد ۶: میزان دقت مدل softmax

با توجه به اندازه گیری های انجام شده به خوبی مشاهده می‌شود که عملکرد الگوریتم softmax از سایر الگوریتم ها بهتر بوده است و بعد از آن الگوریتم OvO قرار دارد که عملکردی مشابه با softmax داشته. البته زمان اجرای این الگوریتم بسیار بیشتر از softmax بوده است. به دلیل آنکه تعداد classifier ها در هر دو حالت OvO و OvR با هم برابرند در نتیجه زمان اجرای دو الگوریتم با یکدیگر تفاوت چندانی ندارند و به نظر می‌رسد دلیل عملکرد ضعیف‌تر روش OvR، نامتوازن بودن تعداد داده‌های مربوط به هر کلاس در مراحل مختلف آموزش مدل است.