نام و نام خانوادگی: حسنا اویارحسینی	شماره دانشجویی: ۹۸۲۳۰۱۰
نام درس: هوش محاسباتی	گزارش پروژه شماره ۱

سوالات داخل متن:

- **قدم دوم (محاسبه خروجی):**

دقت مدل برای ۱۰۰ داده ورودی:

```
def feed_forward(A0, param):
    W1, W2, W3, b1, b2, b3 = param
    Z1, A1 = linear_activation_forward(A0, W1, b1)
    Z2, A2 = linear_activation_forward(A1, W2, b2)
    Z3, A3 = linear_activation_forward(A2, W3, b3)
    return A3, Z3, A2, Z2, A1, Z1
```

```
def calculate_accuracy(train_set, n, param):
    count = 0

    for train_data in train_set[:n]:
        A0 = train_data[0]
        A3, *_ = feed_forward(A0, param)

        predicted_number = np.argmax(A3)
        real_number = np.argmax(train_data[1])

        if predicted_number == real_number:
            count += 1

    return f"{count / n * 100}%"
```

```
param = init_W_b(layers_dim)
print(f"Accuracy: {calculate_accuracy(train_set, 100, param)}")
```

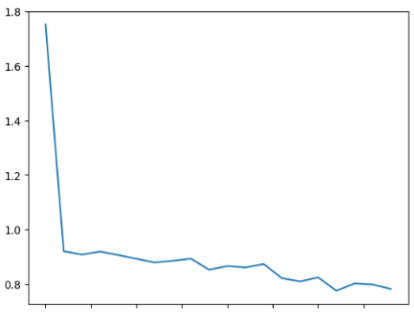
```
Accuracy: 9.0%
```

- **قدم سوم (پیاده سازی back propagation)**

دقت و زمان اجرای یادگیری برای ۱۰۰ داده ورودی: accuracy = ۳۹٪, total time = ۱۴ min

میانگین cost نمونه ها در هر epoch و نمودار مربوط به آن:

```
[28]: %%time
      param, cost_per_epoch = train_non_vectorized(layers_dim, train_set[0: 100], epoch, batch_size, learning_rate)  # this W,b after learning
      CPU times: user 9min 8s, sys: 5min 13s, total: 14min 21s
      Wall time: 4min 59s

[29]: accuracy = calculate_accuracy(train_set, 100, param)
      print("Accuracy :", accuracy)
      Accuracy : 39.0%

[30]: epoch_size = [x for x in range(epoch)]
      plt.plot(epoch_size, cost_per_epoch)

[30]: [<matplotlib.lines.Line2D at 0x7f11243268f0>]
```



```
pprint(cost_per_epoch)
```

```
[array([1.75170128]),
 array([0.92050833]),
 array([0.9084546]),
 array([0.91932788]),
 array([0.9068095]),
 array([0.89301726]),
 array([0.87944025]),
 array([0.88570775]),
 array([0.89338213]),
 array([0.85284323]),
 array([0.86619475]),
 array([0.86141534]),
 array([0.87382117]),
 array([0.8219743]),
 array([0.81021684]),
 array([0.82474697]),
 array([0.77580985]),
 array([0.8027095]),
 array([0.79893054]),
 array([0.78222264])]
```

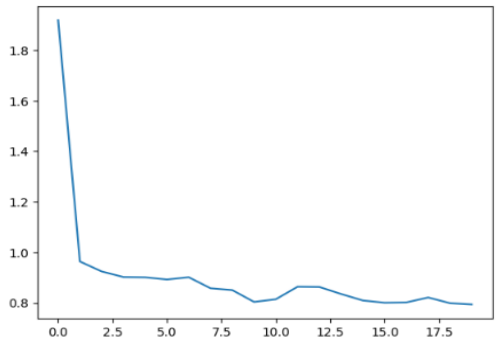
← Average of costs per epoch

- **قدم چهارم (vectorization):**

دقت و زمان اجرای یادگیری برای ۱۰۰ داده ورودی: accuracy = ۴۱٪, total time = ۶,۱۶ s

میانگین cost نمونه ها در هر epoch و نمودار مربوط به آن:

```
[35]: %%time
      param, cost_per_epoch = train_vectorized(layers_dim, train_set[0: 100], epoch, batch_size, learning_rate, vectorized=True)  # this W,b after learning
      CPU times: user 3.72 s, sys: 2.44 s, total: 6.16 s
      Wall time: 1.65 s
[36]: accuracy = calculate_accuracy(train_set, 100, param)
      print("Accuracy :", accuracy)
      Accuracy : 41.0%
[37]: epoch_size = [x for x in range(epoch)]
      plt.plot(epoch_size, cost_per_epoch)
[37]: [<matplotlib.lines.Line2D at 0x7f10edf5b1f0>]
```

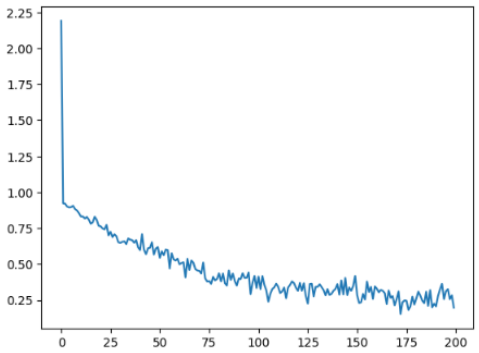


دقت و زمان اجرای یادگیری برای ۱۰۰ داده ورودی با epcoh = ۲۰۰: accuracy = ۹۳٪, total time = ۱min, ۱۲s

s

میانگین cost نمونه ها در هر epoch و نمودار مربوط به آن:

```
[39]: %%time
param, cost_per_epoch = train_vectorized(layers_dim, train_set[:100], epoch, batch_size, learning_rate, vectorized=True)  # this W,b after learning

CPU times: user 41 s, sys: 31.3 s, total: 1min 12s
Wall time: 20.8 s

[40]: accuracy = calculate_accuracy(train_set, 100, param)
print("Accuracy :", accuracy)

Accuracy : 93.0%

[41]: epoch_size = [x for x in range(epoch)]
plt.plot(epoch_size, cost_per_epoch)

[41]: [<matplotlib.lines.Line2D at 0x7f10ede3df60>]
```

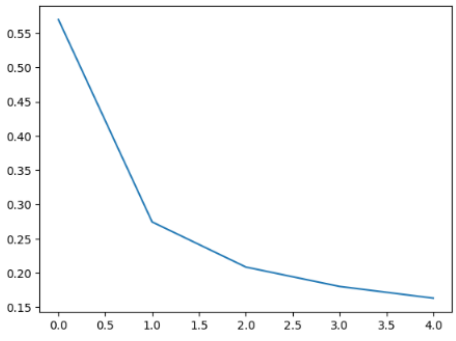


- **قدم پنجم (تست کردن مدل):**

دقت مدل برای مجموعه train: ٪۹۳

نمودار میانگین cost نمونه ها در هر epoch برای مجموعه داده train:

```
ynb
Handwritten Digit Recognitio
Code
Python 3 (ipykernel)

]: # Hyperparameters
batch_size = 50
learning_rate = 1
epoch = 5

]: param, cost_per_epoch = train_vectorized(layers_dim, train_set, epoch, batch_size, learning_rate, vectorized=True)
accuracy = calculate_accuracy(train_set, len(train_set), param)
print("Accuracy :", accuracy)
epoch_size = [x for x in range(epoch)]
plt.plot(epoch_size, cost_per_epoch)

Accuracy : 90.255%

]: [<matplotlib.lines.Line2D at 0x7f10ed6e5060>]
```



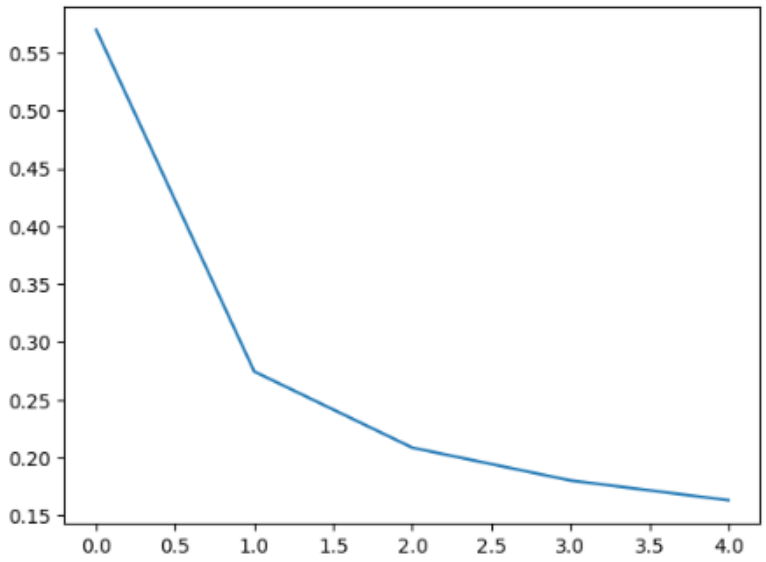
دقت مدل برای مجموعه test: ٪۸۹,۸

نمودار میانگین cost نمونه ها در هر epoch برای مجموعه داده test:

```
accuracy = calculate_accuracy(test_set, len(test_set), param)
print("Accuracy :", accuracy)
epoch_size = [x for x in range(epoch)]
plt.plot(epoch_size, cost_per_epoch)
```

```
Accuracy : 89.8%
```

```
[<matplotlib.lines.Line2D at 0x7f10ed53a8f0>]
```



سوالات بخش تحقیقی امتیازی:

- **سوال اول) هدف از داشتن مجموعه داده ی اعتبارسنجی چیست و چه فرقی با مجموعه داده ی تست دارد؟**

هدف از داشتن مجموعه داده اعتبار سنجی اعتبار سنجی مدلی است که به کمک داده های آموزش آموزش داده شده است اما داده های اعتبار سنجی را ندیده است. در واقع این مجموعه به ما کمک میکند تا دقت مدل را تخمین زده و در صورت نیز بهبود ببخشیم و همچنین این داده ها کمک میکند تا hyperparameter های مدل را بتوانیم بهبود دهیم.

برای مجموعه اعتبارسنجی ما مقدار خروجی (در یادگیری با سرپرستی همان لیبل ها) را میدانیم و به کمک آن تلاش میکنیم مدل را ارزیابی و بهبود ببخشیم اما مجموعه تست به نوعی دست کاربر می باشد و صرفا میزان دقت نهایی مدل را تعیین میکند و از آن نمیتوان برای بهبود مدل ساتفاده کرد. در واقع مجموعه اعتبار سنجی برای بهینه سازی پارامترهای مدل استفاده می شود در حالی که مجموعه تست برای ارائه یک تخمین بی طرفانه از مدل نهایی استفاده می شود.

- **سوال دوم) دو ورژن پیشرفته گرادیان کاهشی گرادیان کاهشی دسته ای**

گرادیان کاهشی تصادفی و گرادیان کاهشی دسته ای کوچک

نام دارددر مورد هر کدام تحقیق کنید و مزایا و معایب هرکدام را بیان کنید.

- گرادیان کاهشی دسته ای: میانگین گرادیان تمام مثال‌های آموزشی را می‌گیریم و سپس از آن گرادیان میانگین برای به‌روزرسانی پارامترهایمان استفاده می‌کنیم. بنابراین این فقط یک مرحله از شیب نزول در یک دوره است. این روش برای توابع که انحنای آن ها نسبتا صاف عالی است.

  مزایا: فهم ساده تر - گرادیان خطای پایدار و همگرایی پایدار / معایب: محاسبات بالا - کند و نیازمن

- گرادیان کاهشی تصادفی : SGD به طور تصادفی یک نقطه داده را از کل مجموعه داده در هر تکرار انتخاب می کند تا محاسبات را بسیار کاهش دهد. این روش برای زمانی که تعداد داده های آموزشی بسیار زیاد ست منیب است.

  مزایا: سرعت بالاتر - کاهش افزونگی در محاسبات داده تکراری - وزن ها را در لحظه انجام محاسبه میتوان به روز رسانی کرد چون فقط یک داده را در نظر داریم / معایب: تابع هزینه به علت تصادفی بودن این روند به شدت نوسان دارد. به مقیاس بندی ویژگی حساس است.

- گرادیان کاهشی دسته ای کوچک: از مجموعه‌ای از تعداد ثابت نمونه‌های آموزشی استفاده می‌کنیم که کمتر از مجموعه داده واقعی است و آن را یک دسته کوچک می‌نامیم و به کمک گرادیان روی این دسته کوچک پارامتر ها را آپدیت میکنیم. این روش سعی می کند بین خوبی گرادیان کاهشی دسته ای و سرعت SGD تعادل ایجاد کند.

  مزایا: واریانس به روز رسانی پارامتر را کاهش می دهد و در نتیجه منجر به همگرایی پایدار می شود. سرعت بهتری از گرادیان دسته ای دارد. / معایب: با توجه به اینکه loss برای هر دسته باید محاسبه شود و میانگین گرفته شود پیاده سازی پیچیده تر است و در لحظه نمیتوان وزن ها را به روز رسانی کرد.

- **سوال سوم) نرمال سازی دسته ای:**

نرمال سازی دسته ای روشی برای بهبود عملکرد و پایداری شبکه های عصبی است. ایده این کار نرمال سازی ورودی هر لایه با کمک میانگین و واریانس می باشد. به شکلی که میانگین اعداد بردار نهایی برابر با صفر و واریانس آن ها برابر با یک شود. با اینکار داده ها به سمت مبدا حرکت می کنند. همچنین تغییر واریانس داده ها به یک هم موجب توزیع یکنواخت داده ها در همان قسمت می شود. با این کار با نرمال سازی داده ها، learning rate در جهت همه ویژگی ها برابر می شود پس در نتیجه یادگیری و بهینه سازی پارامتر ها سریع تر انجا میشود. در حالی که training iteration به علت اضافه شدن سربار محاسباتی نرمال سازی مدت زمان بیشتری را صرف می کند، اما چون به طور معمول به همگرایی زودتر شبکه کمک می کند، پس در مجموع موجب افزایش سرعت خواهد شد.

- **سوال چهارم) در شبکه های کانولوشنیCNN وظیفه لایه Poolingرا مختصر توضیح دهید و روشهای مورد استفاده آن را نام ببرید؛ در انتها با ذکر دلیل توضیح دهید چرا برای مسائل دسته بندی ای که با عکس سر و کار دارد، شبکه های کانولوشنی بر شبکه های ساده ارجحیت دارند؟**

لایه‌ی ادغام با کوچک‌کردن اندازه‌ی عکس ورودی و خلاصه‌سازی ویژگی‌های اصلی و مهم موجود در عکس، به کاهش محاسبات شبکه و مشکل Overfitting کمک می‌کند. در واقع بعد از هر بار عمل کانولوشن که در خروجی به ما یک

نقشه‌ی ویژگی (Feature Map) می‌دهد یک بار فرایند ادغام (Pooling Layer) انجام می‌شود که این لایه باعث میشود:

۱- اندازه‌ی نقشه‌های ویژگی (Feature Maps) را کاهش می‌دهد که این خود تعداد پارامترهای یادگیری و هم‌چنین میزان محاسبات شبکه را کاهش می‌دهد.

۲- ویژگی‌های موجود در هر ناحیه از نقشه‌ی ویژگی را که لایه‌ی کانولوشن تولید کرده است خلاصه می‌کند؛ درواقع مهم‌ترین آن‌ها را انتخاب می‌کند و به مرحله‌ی بعد منتقل می‌کند. این باعث می‌شود مدل در برابر تغییرات موقعیت ویژگی‌های موجود در تصویر ورودی مقاومت بیشتری داشته باشد.

انواع لایه ادغام:

- Max pooling: بزرگ‌ترین مقدار در ناحیه‌ای را که فیلتر پوشانده است انتخاب می‌شود؛ بنابراین در این حالت خروجی یک نقشه‌ی ویژگی (Feature Map) است که برجسته‌ترین ویژگی‌های نقشه ویژگی (Feature Map) قبلی را دارد.
- Average pooling: میانگین ناحیه‌ای که فیلتر روی آن قرار می‌گیرد محاسبه می‌شود؛ بنابراین میانگین ویژگی‌های نقشه‌ی ویژگی قبلی را در خروجی ارائه می‌کند.
- Sum pooling: جمع کل ناحیه‌ای که فیلتر پوشانده است محاسبه می‌شود و یک نقشه‌ی ویژگی جدید را ایجاد می‌کند.

با توجه به اینکه ورودی عکس شامل داده های زیادی به ازای هر عکس میباشد استفاده از شبکه کانولوشنی بر شبکه ساده ارجحیت دارد زیرا شبکه کانولوشنی تعداد واحدهای شبکه را کاهش میدهد به این معنی که به پارامترهای کمتری برای یادگیری نیاز دارد و احتمال overfit نیز کم میشود. اما اگر میخداستیم از شبکه ساده استفاده کنیم حجم محاسابت و داده ها بسیار افزایش میافت.