

Μηχανική Μάθηση - Αναφορά
1ης προγραμματιστικής εργασίας
Δανοπούλου Αιμιλία 3170033

Υπολογισμός Παραγώγου του W1

Για τον υπολογισμό του w1 χρησιμοποιήθηκε ο κανόνας της αλυσίδας και πηγαίνοντας προς τα πίσω κάνουμε τους αντίστοιχους υπολογισμούς.

Υπολογισμός παραγώγου του w1
Chain rule

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y_{nk}} \cdot \frac{\partial y_{nk}}{\partial a_{nk}} \cdot \frac{\partial a_{nk}}{\partial z_n} \cdot \frac{\partial z_n}{\partial x} \cdot \frac{\partial x}{\partial w_1}$$

οπου $a_{nk} = w_2^T \cdot z_n$

~~y_{nk}~~ = ~~t_{nk}~~ , ~~a_{nk}~~

$$\frac{\partial E}{\partial a_{nk}} = t_{nk} - y_{nk} , \frac{\partial a_{nk}}{\partial z_n} = \frac{\partial w_2^T \cdot z_n}{\partial z_n} = w_2$$
$$\frac{\partial z_n}{\partial x_b} = \frac{\partial h(x_b)}{\partial x_b} = h'(x)$$

$$\text{onou } \alpha_{nk} = w_2^T \cdot z_n$$

$$\cancel{\alpha_{nk}} = \cancel{t_{nk}}, \quad \cancel{\alpha_{nk}}$$

$$\frac{\partial E}{\partial \alpha_{nk}} = t_{nk} - y_{nk}, \quad \frac{\partial \alpha_{nk}}{\partial z_n} = \frac{\partial w_2^T \cdot z_n}{\partial z_n} = w_2$$

$$\frac{\partial z_n}{\partial x_b} = \frac{\partial h(x_b)}{\partial x_b} = h'(x)$$

$$\frac{\partial x_b}{\partial w_1} = \frac{\partial x_b \cdot w_1^T}{\partial w_1} = x_b$$

$M \times$

$-T$

\uparrow

$$\frac{\partial x_b}{\partial x_b} = \frac{\partial x_b}{\partial x_b} =$$

$$\frac{\partial x_b}{\partial w_1} = \frac{\partial x_b \cdot w_1^T}{\partial w_1} = x_b$$

$M \times (D+1)$

$$\frac{\partial E}{\partial w_1} = \left[(t - y) \cdot w_2 \cdot h'(x) \right]^T \cdot x_b - \lambda \cdot w_1$$

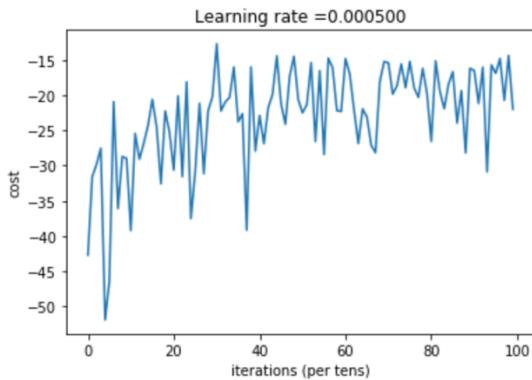
$\underbrace{\quad}_{N_b \times K}$ $\underbrace{\quad}_{K \times M}$ $\underbrace{\quad}_{N_b \times M}$ $\underbrace{\quad}_{N_b \times (D+1)}$
 $N_b \times M \rightsquigarrow M \times N_b \rightsquigarrow M \times N_b \cdot N_b \times (D+1) \Rightarrow$
 $M \times (D+1)$
 $\mu \varepsilon \delta \sigma \pi \rho \alpha \gamma \omega \sigma$
 w_1

Παρακάτω ακολουθούν screenshots με τα αποτελέσματα της εκτέλεσης του ml_train για τις διάφορες τιμές του M = 100, 200, 300 και για κάθε συνάρτηση.

1. M =100 για την πρώτη συνάρτηση log με mini-batch =100

```
: plt.plot(np.squeeze(ncost))
plt.ylabel('cost')
plt.xlabel('iterations (per tens)')

plt.title("Learning rate =" + str(format(options[2], 'f')))
plt.show()
```

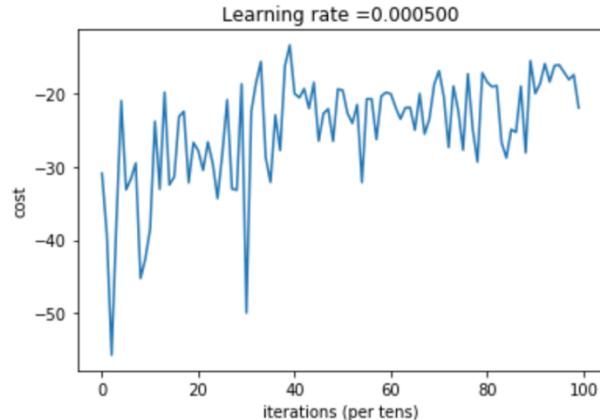


With accuracy

```
np.mean( pred == np.argmax(y_test, 1) )
```

0.9696

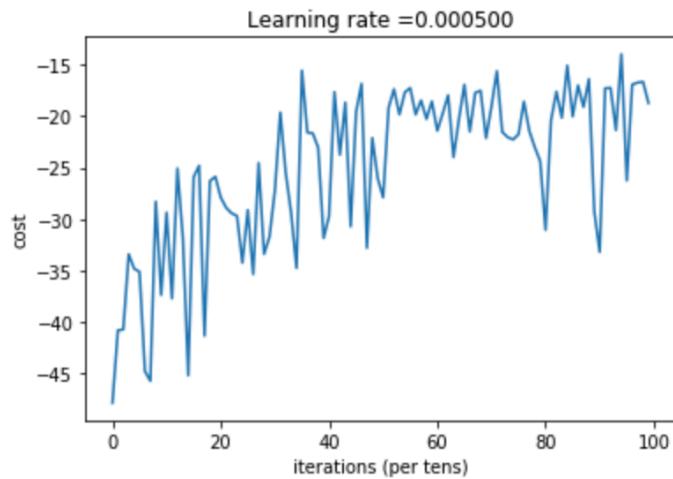
2.M =200 για την πρώτη συνάρτηση log με mini-batch =100



```
np.mean( pred == np.argmax(y_test, 1) )
```

0.9687

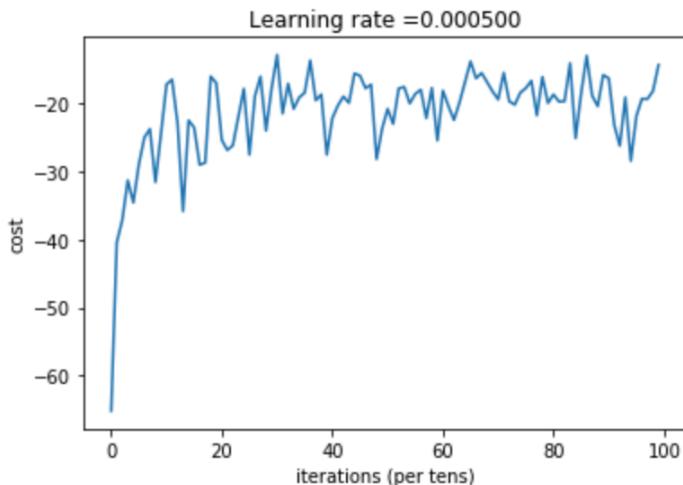
3.M =300 για την πρώτη συνάρτηση log με mini-batch =100



```
np.mean( pred == np.argmax(y_test,1) )
```

0.97

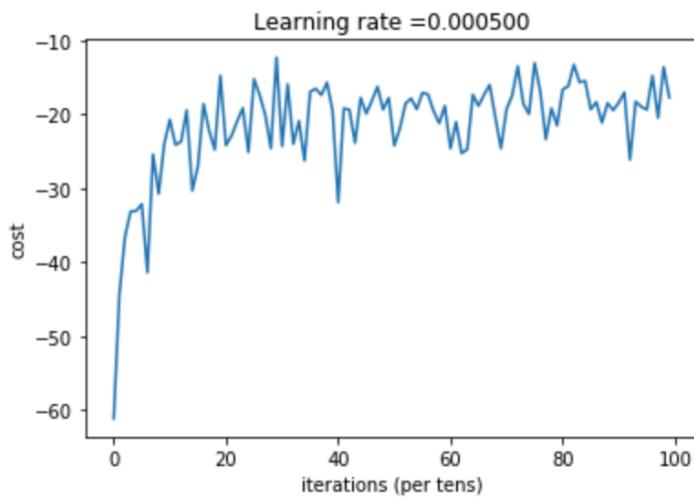
4.M =100 για την δεύτερη συνάρτηση με mini-batch =100



```
np.mean( pred == np.argmax(y_test,1) )
```

0.9711

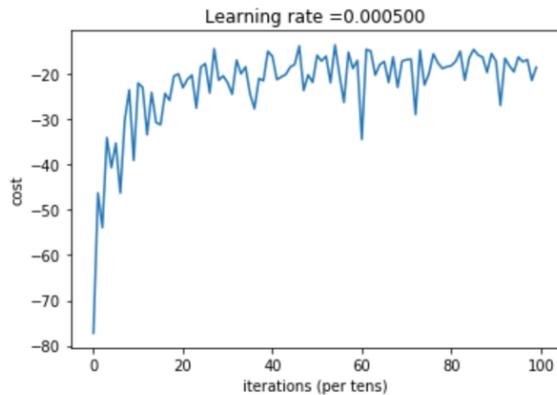
5.M =200 για την δεύτερη συνάρτηση με mini-batch =100



```
np.mean( pred == np.argmax(y_test, 1) )
```

0.9735

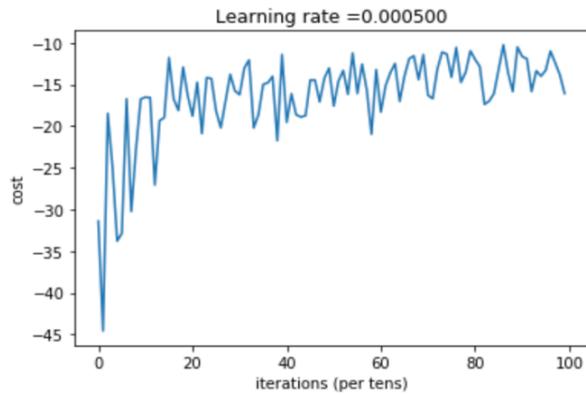
6.M =300 για την δεύτερη συνάρτηση με mini-batch =100



```
np.mean( pred == np.argmax(y_test, 1) )
```

0.9729

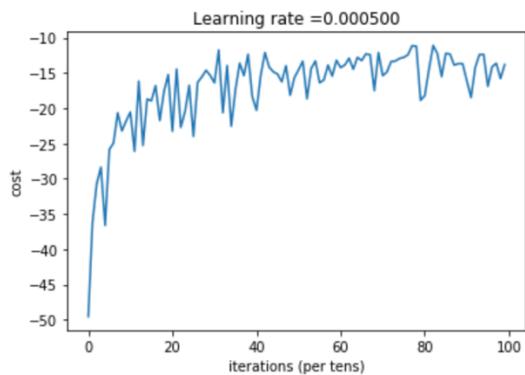
7.M =100 για την τρίτη συνάρτηση με mini-batch =100



```
np.mean( pred == np.argmax(y_test,1) )
```

0.9784

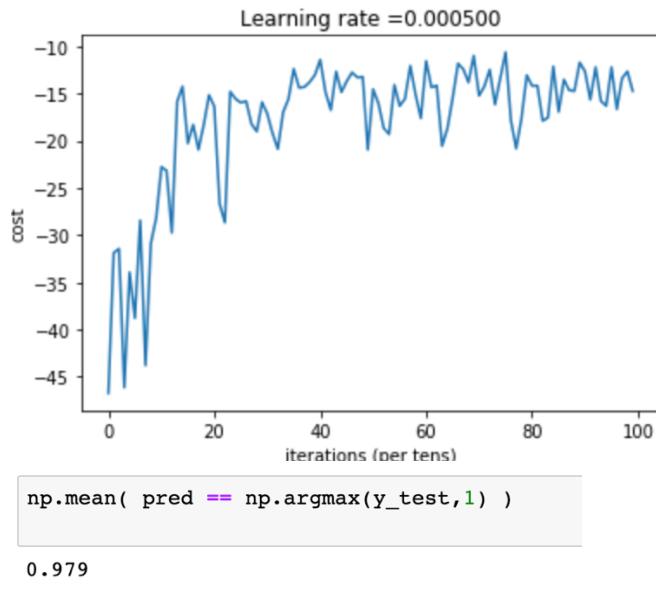
8.M =200 για την τρίτη συνάρτηση με mini-batch =100



```
np.mean( pred == np.argmax(y_test,1) )
```

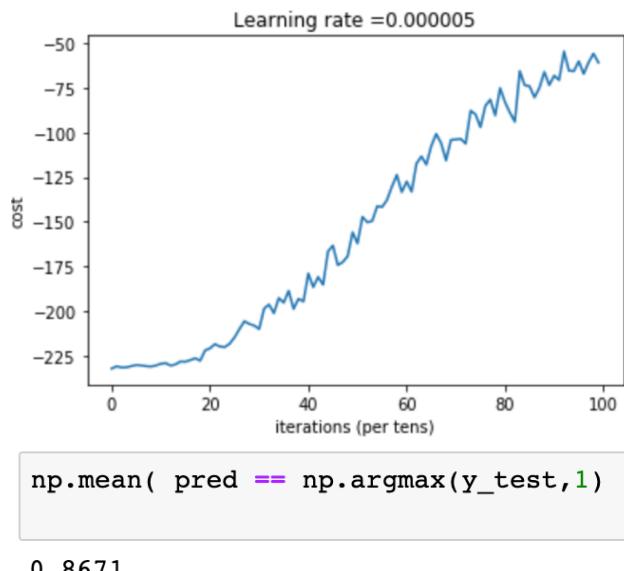
0.9796

9.M =300 για την τρίτη συνάρτηση με mini-batch =100



Δοκιμή διαφορετικού learning rate

Δοκιμή για $M = 100$ χρησιμοποιώντας την δεύτερη συνάρτηση και learning rate = 0.000005

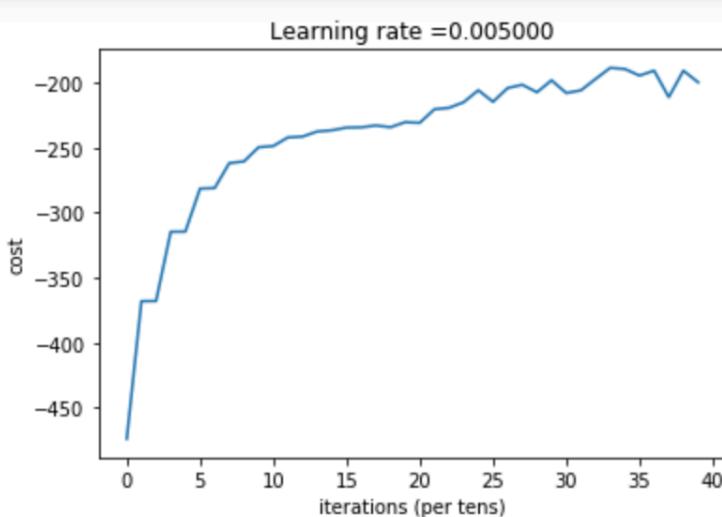


Παρατηρούμε ότι το accuracy δεν είναι όσο καλό όσο πριν. Επιπλέον για μεγάλο learning rate πχ 0.5 υπήρχαν overflows στον υπολογισμό των activation functions.

Cifar – 10 dataset

Με σκοπό να τρέξει καλύτερα η συνάρτηση ml_train στον υπολογιστή μου μετέτρεψα τις εικόνες σε grayscale 50.000 x 1024 και απέφυγα κάποια overflows που είχα για τις 2 πρώτες συναρτήσεις όταν οι εικόνες ήταν έγχρωμες.

Το Cifar dataset για 40 εποχές και με την χρήση της πρώτης συνάρτησης είχε την εξής παράσταση:

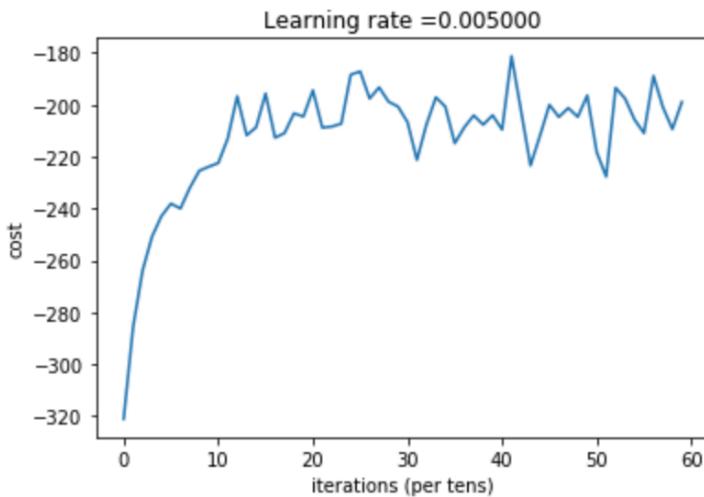


```
pred = ml_softmax_test(d1,e1, x_test,choice)
np.mean( pred == np.argmax(y2,1) )
```

0.3123

Με accuracy: 0.3123

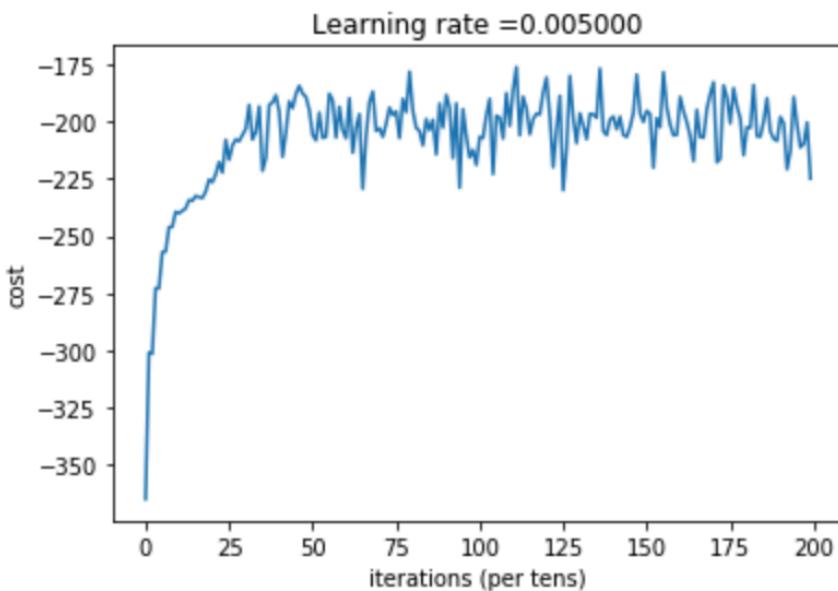
Για **M = 200 , 60 εποχές και την πρώτη συνάρτηση**



```
pred = ml_softmax_test(updated_w1,updated_w2, x_test,choice)
np.mean( pred == np.argmax(y2,1) )
```

0.3197

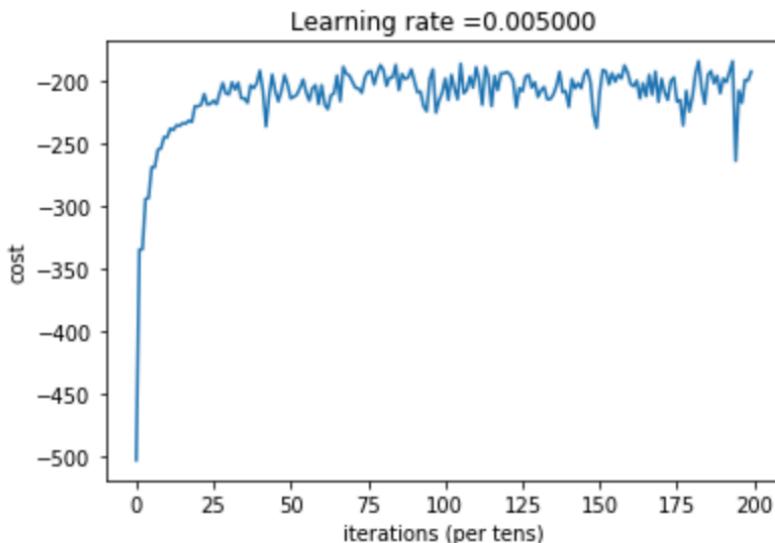
Για $M = 200$, για την πρώτη συνάρτηση και για 100 εποχές.



```
pred = ml_softmax_test(updated_w1,updated_w2, x_test,choice)
np.mean( pred == np.argmax(y2,1) )
```

0.3259

Για $M = 100$ και με την πρώτη συνάρτηση για 100 εποχές.



```
pred = ml_test(updated_w1,updated_w2, x_test,choice)
print('Accuracy is ',np.mean( pred == np.argmax(y2,1) ))
```

Accuracy is 0.3014

Για τις δύο άλλες συναρτήσεις exp και cos, το accuracy βγαίνει χαμηλό, δηλαδή περίπου 10%. Η παράμετρος lamda για το Cifar ήταν 0.1 σε όλα τα παραπάνω παραδείγματα.