יובל פרידמן 204169320

תשובות וקוונטיזציה תר׳ 6

1

1.1

$$\ln(g(y)) = \ln\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\mu)^2}{2\sigma^2}}\right) = \ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + \ln\left(e^{-\frac{(y-\mu)^2}{2\sigma^2}}\right) =$$

$$= \ln(1) - \ln(\sqrt{2\pi}) - \ln(\sigma) - \frac{(y-\mu)^2}{2\sigma^2}$$

$$\Rightarrow e_\sigma = \frac{\partial \ln(g(y))}{\partial \sigma} = -\frac{1}{\sigma} - \frac{(y-\mu)^2}{\cancel{2}} \cdot \left(-\frac{1}{\sigma^3}\right) \cdot \cancel{2} =$$

$$-\frac{1}{\sigma} + \frac{(y-\mu)^2}{\sigma^3}$$

$$\Rightarrow \Delta\sigma = \eta \cdot R \cdot e_\sigma = \eta \cdot R\left(\frac{(y-\mu)^2}{\sigma^3} - \frac{1}{\sigma}\right)$$

כשהרעש בדגימות גדול, אז פוחתת המגמות של מרחק הרחבאים של הנתון $(y-\mu)^2$ מהמקום אליו הם כיוון, אך ככל שהרעש נעשה קטן יותר, אז השינוי ברעש מושפע לא ממגמת הרעש אלא ממרחק בהרכאה.

1.2

$$E[\Delta\sigma] \underset{\text{משפט לינאריות התוחלת}}{=} \eta \cdot \frac{\partial}{\partial\sigma} E[R]$$

השינוי המצופה ב-σ:

$$E[R] = E[2hy - y^2 - h^2] = 2hE(y) - E[y^2] - h^2$$

$$E(y) = \mu \, , \quad E[y^2] = var(y) + E^2(y) = \sigma^2 + \mu^2$$

$$\Rightarrow E[R] = 2h\mu - \sigma^2 - \mu^2 - h^2$$

$$\Rightarrow \frac{\partial}{\partial\sigma} E[R] = -2\sigma$$

$$\Rightarrow E[\Delta\sigma] = -2\eta\sigma$$

ההתכנסות במודל משיגת כאשר:

$$E[\Delta\sigma] = -2\eta\sigma = 0$$

$$\Rightarrow \sigma = 0$$

רק כאשר הלמידה תתכנס הצריקות כולן לא יהיו רועשות, כלומר הלמידה הרבה תרגש.

(2)

(2.1)

$$e_i(y=1) = \frac{\partial \ln(P(y=1))}{\partial w_i} = \frac{\partial}{\partial w_i} \ln\left([1+e^{-w^T x}]^{-1}\right) =$$

$$-\frac{1}{1+e^{-w^T x}} \cdot e^{-w^T x} \cdot (-x_i) = P(y=0)\, x_i = [1-P(y=1)]\, x_i$$

$$e_i(y=0) = \frac{\partial \ln(P(y=0))}{\partial w_i} = \frac{\partial}{\partial w_i} \ln\left(\frac{1}{e^{w^T x}+1}\right) =$$

$$-\frac{1}{e^{w^T x}+1} \cdot e^{w^T x} \cdot x_i = -\frac{1}{1+e^{-w^T x}}\, x_i = -P(y=1)\, x_i$$

(2.2)

$$e_i(Y=y) = [y - P(y=1)]\, x_i$$

$$y \in \{0,1\}$$

(2.3)

$$\Delta w_i = \eta \cdot R \cdot e_i = \eta \cdot R [y - P(y=1)]\, x_i$$

אם אנחנו מתייחסים אל ההסתברויות כנתונות מראש לפני כל עדכון לוקאלי, כלומר שבכל פעם $w^T x$ כבר ידוע, אזי בשאלה הזאת הכלל ״לוקאלי״ תלוי רק בגלל R: אם R תלוי בסינפסה w_j $(j \neq i)$ ואחרת אז הכלל לא לוקאלי אבל אחרת הוא כן.

אולם, אם אנחנו מתייחסים אל ההסתברות כאילו נתונה מראש, כלומר שהיא מחושבת על פי ערכי הסינפסות האחרות עבור כל סינפסה בנפרד, אזי ככה נגיע לכך שאנחנו מסוגלים להבין שכלל העדכון כולו אינו לוקאלי.

Scanned by CamScanner

In [57]:
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
ll drive.mount("/content/drive", force_remount=True).
```

In [0]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
```

###Q1

In [0]:
```python
# Python Dict, "data", "test_data", "labels" and "test_labels" are keys
matlab_file = loadmat("/content/drive/My Drive/C&C/ex6/ex6_data.mat")

# Define train and test examples and labels
X_train, X_test, y_train, y_test = matlab_file['data'], matlab_file['tes
matlab_file['labels'].reshape(-1), matlab_file['test_labels'].reshape(-1
```

In [60]:
```python
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(784, 12665)
(784, 2115)
(12665,)
(2115,)
```

###Q3 Done before Q2 as we want to break by this functions the learning of Q2.

In [0]:
```
''' Fucntion for sampling output from vector of probabilities to be equa
  p1: vector of p1's according to test examples
  '''
def sample_output(p1):
  generator = np.vectorize(lambda p: np.random.choice([0,1],p=[1-p,p]))
  return generator(p1)


''' Function for testing the accuracy of a weights vector on test exampl
  w: current weights vector
  X: test examples
  labels: labels of test examples
  '''
def test_accuracy(w,X,labels):
  wTx = np.apply_along_axis(lambda example: np.dot(example,w), axis=0,ar
  p1 = 1/(1+np.exp(-wTx))
  y = sample_output(p1=p1)
  return np.sum(y == labels) / X.shape[1]
```

###Q2

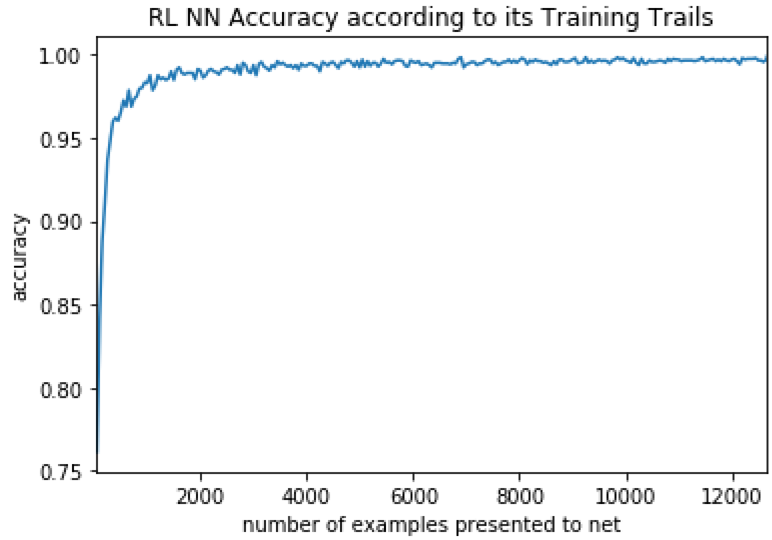
In [0]:
```
ETA = 0.01
w = np.random.normal(0,0.01, size=X_train.shape[0])  # initialization

breaks = np.append(np.arange(start=49, stop=X_train.shape[1], step=50),[
experiment = pd.Series(index=breaks)
for j in range(X_train.shape[1]):
  x = X_train[:,j]
  p1 = 1/(1+np.exp(-np.dot(w,x)))  # p(y=1)
  p0 = 1-p1  # p(y=0)
  y = np.random.choice([0,1],p=[p0,p1])
  r = (y == y_train[j]).astype(int)  # reward of y
  e_w = (y-p1)*x  # eligibility w.r.t the vec w
  w += ETA*r*e_w
  if j in breaks:
    experiment[j] = test_accuracy(w=w,X=X_test,labels=y_test)
```

###Q4

In [63]:

```
experiment.plot()
plt.title("RL NN Accuracy according to its Training Trails")
plt.xlabel("number of examples presented to net")
plt.ylabel("accuracy")
plt.show()
```

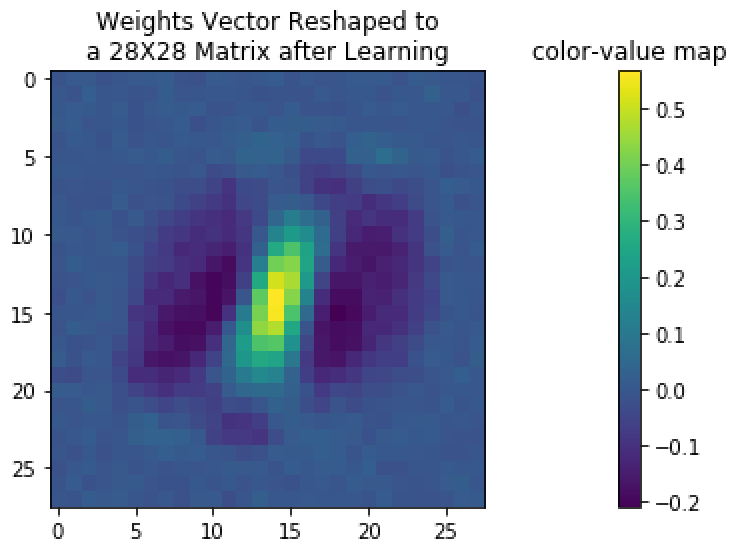


We can be impressed that after an approximately 700 examples the net achieved good results above the test examples.

###Q5

In [64]:

```
plt.imshow(w.reshape(28,28))
plt.colorbar(pad=0.2).ax.set_title("color-value map")
plt.title("Weights Vector Reshaped to\na 28X28 Matrix after Learning")
plt.show()
```



It seems that the colors around the digit 1 are contrary to the colors in the area of the digit 0. This picture is a consequence of the way 0\1 digits of the net's learning and the way MNIST data encodes the images. The MNIST encodes the black part of its images as low values (close to 0) and its white part (the places where the digits are written in the image) as high values (close to 1). The learning produces a single weights vector which has to classify every image which will presented to it, so the learning assigns certain weights to the area that approximately corresponding to the ones area in the input vectors, and a cotrary weights to the area that approximately corresponding to the zeros area in the input vectors. As a result, we get the above picture.