Itay Hassid, 209127596 Aviv Shisman, 206558157

AML Ex2

comparison later, look at readMe please

First model

The first model we used is a standard multiclass perceptron,
We used k classifiers in size 1*128 each, where each W[i] had an prediction and we took the max one.
(the prediction was a standard np.dot(w,x) like previous exercises in ML)

As for the update rule we used the one in the appendix,if j (the class of the current example) is not equal to ytag (the class we predicted) for each w[i]:
Increase the classifier that classify class j
Decrease the classifier that classify class ytag
That way we increase the chance that next time it will classify it as class j and decrease the chance of classifying it as class ytag.

Our w in the test is the average of all the w's (from every wi) from all the epochs and calculated in the end of the train.
Also we shuffled the train set in the beginning of every epoch.

Accuracy: 73.37674418604651%

```
itay@itay-Vostro-5468:~/Documents/AML/ex2$
ain.data data/letters.test.data
LoadInfo...
Model 1 Training...
prediction...
Accuracy: 73.37674418604651%
itay@itay-Vostro-5468:~/Documents/AML/ex2$
```

cont next page

Second Model

In the second model we created a structured perceptron from the multiclass model,
This time we have one classifier in size 1,26*128 , the prediction is same(w*x) as before except that this time x is a zero vector in size 26*128,1 and the pixels are in the yi place at the vector.
Its looks like (0 0 0 …. Pixels 0 0 0) where pixels start in index 128*k (k represents index of letter between 0 to 25, 26 letters in english).

As for the update rule we used : (if y!=ytag)
w+= phi(x,y) - phi (x,ytag)
That way as before, we will increase the chance of classifying it as y and decrease the chance of classifying it as ytag.

We did the same averaging method in the second model (calculated after training ended) and shuffled the train in every epoch.

Accuracy: 73.56279069767442%

```
itay@itay-Vostro-5468:~/Documents/AML/ex2$
ain.data data/letters.test.data
LoadInfo...
Model 2 Training...
prediction...
Accuracy: 73.56279069767442%
itay@itay-Vostro-5468:~/Documents/AML/ex2$
```

cont next page

Itay Hassid, 209127596 Aviv Shisman, 206558157

Third model

In this model we will count the previous label of each example as a feature, in order to do that we will use dynamic programing , and for that we take the letters of the train set and chunk them into words (the last letter in the train set has '-1' in its field).
Our classifier in this model is similar to the second model, it's size is 1,(26*128 + 27*27), the additional 27*27 are all the possible bi-grams.

In every epoch, we iterate over a list of words (in the beginning of each epoch we shuffle the list), call viterbi function(the dynamic algorithm) that returns using predicted tags for the word and then we use the update rule:
(w = w + build_phi(word, y) - build_phi(word, tags))
The update similarly to previous models will increase the wights in a way that will increase the chances of labeling right next time,
notice that the build_phi function will extract a vector with the features at the yth place and the indicators at the yth *y-1th place that way we are increasing the classifier "good weights" and decreasing the "bad weights".

Again, like in the previous models, we did an average on sum of all the w's from all the epochs.

accuracy:
83.49767441860465%
8 epochs, 5 minutes per epoch.

```
prediction...
Accuracy: 83.49767441860465%
itay@itay-Vostro-5468:~/Documents/AML/ex2
```

Itay Hassid, 209127596 Aviv Shisman, 206558157

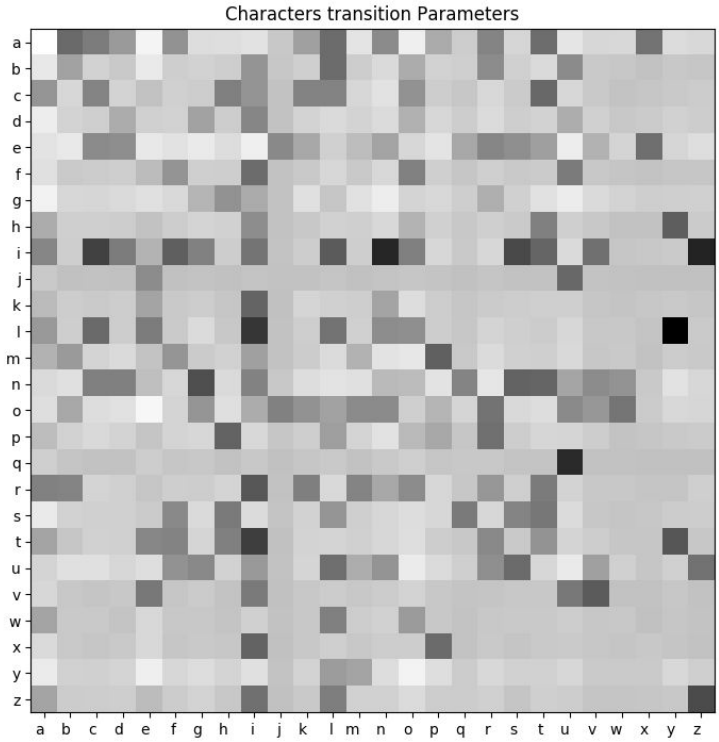
Conclusions:

The 1st and 2nd models achieved around 70% with one epoch which last less than a minute and in few more epochs we reached 73%-74% in few minutes.
In the 3rd model we got around 69% with a single epoch which is lower than the first models, but when we do another epoch we reach to 80% and with few more we got 83%-84%. This model is slower than previouses.

From comparing between the methods, we see that the models 1 & 2 are faster and converge faster but they converge to 74% that relatively to model 3 is low.
Model 3 takes much more time (a few minutes per epoch), but can reach higher results - 84%.

Bonus 1:

The heat-map represent all the possibles bi-grams which is the additional 27*27 cells we added to w but the size of heat-map is 26*26.
We removed the bottom row because it contained the bi-gram of ‘$’ as previous letter and it’s not relevant anymore.
We also remove most right column since it’s represent the bi-gram of (letter, ‘$’) that doesn’t exist so the column was at same color and it’s not relevant either.



Cont next page

Itay Hassid, 209127596 Aviv Shisman, 206558157

<u>Bonus 2:</u>

The main differences between structured perceptron to structured svm:

1. Optimization problem , regularization

$$\underset{w,\xi}{\operatorname{argmin}} \frac{1}{2} w^T w + \frac{C}{N} \sum_i \xi_i$$

2. Different training:

Structural SVM Training

- **STEP 0:** Specify tolerance ε
- **STEP 1:** Solve SVM objective function using only working set of constraints **W** (initially empty). The trained model is w.
- **STEP 2:** Using w, find the y' whose constraint is most violated.
- **STEP 3:** If constraint is violated by more than ε, add it to **W**.

Constraint Violation Formula: $$\left(\frac{1}{M_i} \sum_j 1_{[y'^j \neq y_i^j]} + \xi_i \right) - \left(F(y_i, x_i) - F(y', x_i) \right) \geq \varepsilon$$

- **Repeat STEP 1-3** until no additional constraints are added. Return most recent model w trained in STEP 1.