Kernel density estimation with Mixture of Gaussians

Aritra Chowdhury

April 18, 2018

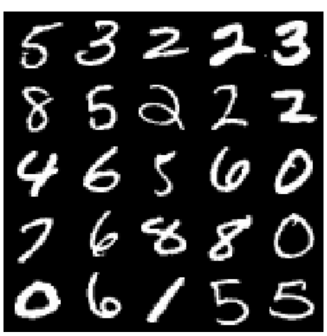
The code for this work has been written in a `Python 3.6.1` environment. Resources for this work is in `enlitic.tar.gz` file. Data, code and results are in the `enlitic/data`, `enlitic/prototypes` and `enlitic/results` directory respectively. Download *MNIST* and *CIFAR* data and extract in `enlitic/data` directory.

1 Pre-processing

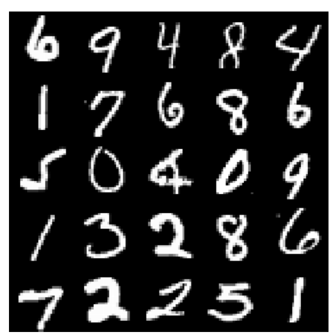
The preprocessing routine may be run in the following from the project directory.

```
python prototypes/preprocessing.py
```

The shuffling and splitting of the *MNIST* data produced the following results.



(a) First 25 training samples



(b) First 25 validation samples

Figure 1: Training and validation samples for *MNIST* dataset

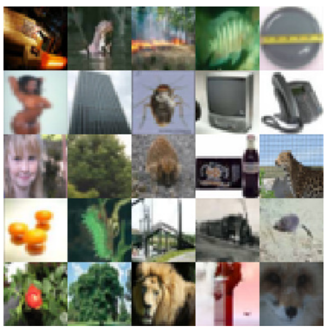
The training and validation labels for the first 25 samples are:

```
MNIST:
Training labels:
```

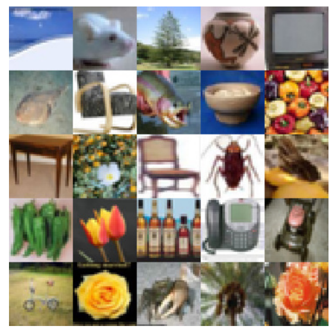
```
[5 3 2 2 3 8 5 2 2 2 4 6 5 6 0 7 6 8 8 0 0 6 1 5 5]
Validation labels:
[6 9 4 8 4 1 7 6 8 6 5 0 4 0 9 1 3 2 8 6 7 2 2 5 1]
```

As we can see, they correspond to Fig. 1

The shuffling and splitting of the *CIFAR* data produced the following results.



(a) First 25 training samples



(b) First 25 validation samples

Figure 2: Training and validation samples for *CIFAR* dataset

The training and validation labels for the first 25 samples are:

```
Training labels:
[sofa_s_000382, otter_s_001595, forest_s_001577,
  cichlid_fish_s_000322, dessert_plate_s_000164,
  black_woman_s_001254, skyscraper_s_002141,
  oriental_cockroach_s_000712, tv_s_000630, phone_s_000206,
  girl_s_000753, pine_tree_s_000542, hedgehog_s_001444,
  soda_bottle_s_001623, leopard_s_001132,
  valencia_orange_s_000231, caterpillar_s_001171,
  drawbridge_s_001730, railroad_train_s_000002,
  water_shrew_s_000236, rosebush_s_001379,
  quercus_robur_s_001230, panthera_leo_s_000894, wtc_s_002139,
  red_fox_s_001215]
Validation labels:
[adriatic_sea_s_001325, mus_musculus_s_001385, pine_tree_s_001775
  , bowl_s_002516, television_s_000971, numbfish_s_000087,
  easy_chair_s_001518, trout_s_000133, fishbowl_s_000687,
  bell_pepper_s_001212, table_s_000156, poppy_s_001633,
  chaise_s_001448, cockroach_s_000146, butterfly_s_003188,
  sweet_pepper_s_002134, tulipa_clusiana_s_000194,
```

```
whiskey_bottle_s_001038 , telephone_s_000577 , mower_s_001046 ,
bicycle_s_000723 , rose_s_001448 , northern_lobster_s_000333 ,
palm_tree_s_001893 , rose_s_001811 ]
```

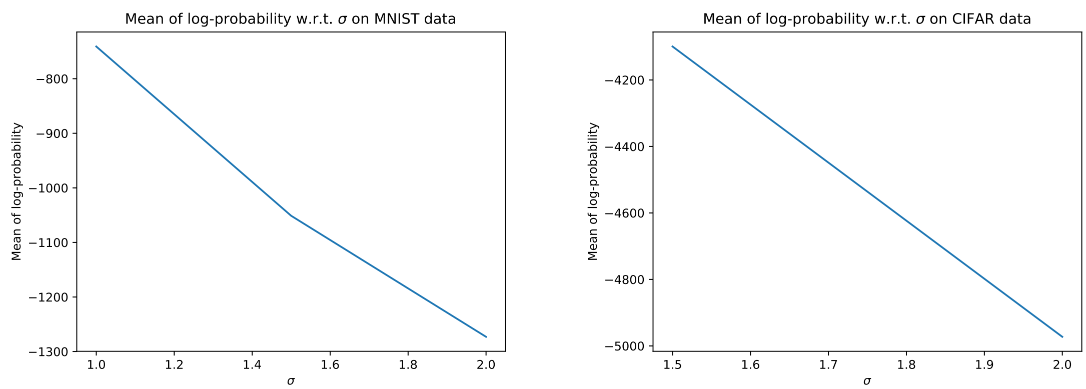
As we can see, they correspond to Fig. 2

2 Computation of log-probability

The computation of log probability routine may be run in the following from the project directory.

```
python prototypes/log_probability.py
```

The grid search plots for *MNIST* and *CIFAR* data is shown in the following figure.



(a) Grid search on *MNIST* data

(b) Grid search on *CIFAR* data

Figure 3: Grid search w.r.t σ on $\mathcal{L}_{D_B}$ of validation data

Fig. 3 shows the plots of the grid search over σ values. The optimal σ values are given by:

$$\sigma^*_{MNIST} = 1, \sigma^*_{CIFAR} = 1.5 \tag{1}$$

We can see a trend in both the plots that there is a clear maximum over the grid points for σ. The σ parameter controls the smoothing of the kernel density estimator. Small values of σ results in noisy estimates of the probability density which results in small values for the log probability and larger values of σ results in over-smoothing and smaller estimates of $\mathcal{L}_{D_B}$.

The log probabilities ($\mathcal{L}_{D_B}$) and runtime on the test data for *MNIST* dataset is:

```
Mean of log probability of test dataset for MNIST data =
    −740.925258217
```

```
Time taken to compute mean of log probability of test dataset for
    MNIST data = 1.7869653701782227 seconds
```

The log probabilities ($\mathcal{L}_{D_B}$) and runtime on the test data for *CIFAR* dataset is:

```
Mean of log probability of test dataset for CIFAR data = nan
Time taken to compute mean of log probability of test dataset for
    CIFAR data = 4.479619264602661 seconds
```