Hackathon

Sarthak Bisht

Resources for HW1P2

- Recitation 1c (Piazza @72): Phoneme Classification Using MLP (toy dataset)
- HW0P2 (Piazza @11): Variations of dataset class for MFCCs (Ex Dataset 3, 4, 5, 6)
- HW1P2 Writeup, Starter Notebook, Suggestions (Recitation 2 & Piazza @118)
- Neural Architecture Search on Toy Dataset (Piazza @221)
- HW1P2 Low Cutoff Architectures (Piazza @261)
  - **Very Low Cutoff : 65% Accuracy**
  - **Low Cutoff : 75% Accuracy**
  - **Medium Cutoff : 85% Accuracy**
  - **High Cutoff : TBD**

The **attached spreadsheet** has a few suggestions for crossing Low Cutoff in the first few epochs. Below are the hyper-parameters for the same:

- Dataset : train-clean-100 with cepstral mean normalization
- Context : 20
- Regularization : Batchnorm after every layer
- Activation : GELU
- Loss : Cross Entropy
- Optimizer : Adam

Memory Efficient Dataset Class

```
length = NotImplemented
T = 0
for i in range(length)):
  mfcc = NotImplemented
  T += mfcc.shape[0]

self.mfccs = np.zeros(
  (T + 2*context, mfcc.shape[1]),
  dtype=np.float32)
self.transcripts = np.zeros((T, ),
  dtype=np.uint8)
```

```
cx, cy = context, 0
for i in range(length):
  mfcc        = NotImplemented
  transcript  = NotImplemented
  # Cepstral Mean Normalization
  # Remove [SOS] and [EOS]
  # Map phoneme (str) to index (int) values
  T_i = mfcc.shape[0]
  self.mfccs[cx:cx+T_i] = mfcc
  self.transcripts[cy:cy+T_i] = transcript
  cx += T_i
  cy += T_i
```

Medium Cutoff Architecture (85%)

Dataset	: train-clean-360 with cepstral mean normalization
Context	: 20
Regularization	: Batchnorm after every layer
Activation	: GELU
Loss	: Cross Entropy
Optimizer	: Adam
Layer Widths	: [1107, 1024, 1024, 1024, 1024, 40]

Medium Cutoff Architecture (85%)

Epoch 1, train acc 81.698%, valid acc 83.402%
train lss 0.549, valid lss 0.491

Epoch 2, train acc 81.717%, valid acc 83.438%
train lss 0.549, valid lss 0.490

Epoch 3, train acc 85.325%, valid acc 84.348%
train lss 0.429, valid lss 0.461

Epoch 4, train acc 86.255%, valid acc 84.695%
train lss 0.398, valid lss 0.452

Epoch 5, train acc 86.767%, valid acc 84.818%
train lss 0.381, valid lss 0.449

Epoch 6, train acc 87.113%, valid acc 84.936%
train lss 0.370, valid lss 0.447

Epoch 7, train acc 87.366%, valid acc 84.980%
train lss 0.362, valid lss 0.450

Epoch 8, train acc 87.563%, valid acc 84.988%
train lss 0.355, valid lss 0.447

Epoch 9, train acc 87.724%, valid acc 85.043%
train lss 0.350, valid lss 0.446

Colab Pro - Rough Estimates

1. toy dataset : less than 1 minute per epoch
2. train-clean-100 : 7 to 14 minutes per epoch
3. train-clean-360 : 20 to 25 minutes per epoch