Machine Learning and having it Deep and Structured
Homework II Report

Group ”SYChienIsGod”

May 8, 2015

1 The Group

1.1 Members

The members of the group ”SYChienIsGod” are

1. 洪培恒, r02943122
2. 曾泓諭, r03943005
3. 柯揚, d03921019
4. 李啓為, r03922054

2 The Algorithm

The algorithm used for all submissions is a Structured Support Vector Machine with Viterbi as inference algorithm. Using slack variables ξ_n, the SSVM optimises

$$\begin{aligned} &\underset{\boldsymbol{w},\xi_n}{\text{minimize}} && \|\boldsymbol{w}\|^2 + C\sum_{n=1}^{\ell}\xi_n \\ &\text{subject to} && \boldsymbol{w}'\Psi(\boldsymbol{x}_n, y_n) - \boldsymbol{w}'\Psi(\boldsymbol{x}_n, y) + \xi_n \geq \Delta(y_n, y), \quad n = 1,\ldots,\ell, \quad \forall y \in \mathcal{Y} \leq b_i,\ i = 1,\ldots,m \end{aligned} \tag{1}$$

The constraints can be reformulated to:

$$\xi_n \geq \Delta(y_n, y) - \boldsymbol{w}'\Psi(\boldsymbol{x}_n, y_n) + \boldsymbol{w}'\Psi(\boldsymbol{x}_n, y), \quad n = 1,\ldots,\ell, \quad \forall y \in \mathcal{Y} \leq b_i,\ i = 1,\ldots,m \tag{2}$$

This shows that we need to provide a lower bound for ξ_n so that none of the constraints pertaining to (x_n, y_n) is violated. Hence, for all possible y, we need to find the maximum that the value on the right hand site of the inequality can attain. Having found the corresponding y, one has to add (x_n, y) to the set of points being optimised as negative example. If no constraint's right hand side exceeds ξ_n, no constraint needs to added.
The standard form of a quadratic optimisation problem is

$$\begin{aligned} &\underset{x}{\text{minimize}} && x^TQx + v^Tx \\ &\text{subject to} && f_i(x) \leq 0 \forall i \end{aligned} \tag{3}$$

The above optimisation problem can be fit by stacking $\boldsymbol{w}$ on top of ξ_n, setting Q to the identity for those dimensions that correspond to $\boldsymbol{w}$ and to 0 for all others. v is set to C for those dimensions corresponding to ξ_n and to 0 elsewhere. After solving the optimisation problem, there may be new constraints that have been violated. These are added to the optimisation set and the QP solver searches for an optimal $\boldsymbol{w}$ and ξ_n again.

3 The Experiments

3.1 Data Splitting

To reduce the divergence between the validation data accuracy estimates and the results on the Kaggle data set, the data was split by finding disjoint subgraphs in the graph connecting speakers and sentences. This way, the validation set would only contain sentences and speakers that are not in the training set. The submitted results showed that even little improvements on the validation data would project to the (public) Kaggle results, hence this strategy yielded a reliable performance forecasting.

3.2 Structured SVM with Viterbi

As baseline for our experiment served the suggested structured Support Vector Machine using the Viterbi algorithm for inference. The results on bare FBANK features were 19.45 for $C = 10$ and 18.25 for $C = 100$. Using the output of a neural network with a width (before softmax) of 80, we obtained 10.07 for $C = 1000$. With a width of 100, this reduced to 9.58. Finally, with respect to the results from the error analysis (see below), we filtered out isolated phonemes (where both adjacent phonemes are different), yielding an improvement to 9.44. All values are based on the public part of the Kaggle result.

3.3 Structured SVM with Higher Order Viterbi

We also examined a higher order Viterbi (e.g. letting the state depend on the previous two states: $p(s_i|s_{i-1}, s_{i-2})$ or the emission depend on the current and the previous state: $p(e_i|s_i, s_{i-1})$). However, modelling the vector w to achieve this requires a state cube (i.e. a 39^3-dimensional state transition tensor) to hold the state transition probabilities or an expansion of the emission part of the vector to $39^2 \cdot N_{Features}$. Both approaches were difficult to get to work as memory and computation requirements rose significantly. A validation score (average edit distance) of 16.0534 was achieved with a cost penalty of $C = 100$ using second order state transitions. Obviously, this did not improve previous results.

3.4 Structured Perceptron

PyStruct's implementation of the Structured Perceptron was evaluated for reference and an average edit distance of 12.14 was achieved after 500 iterations. This is comparable to our SVM Struct implementation when setting $C = 1$ (although it would take more iterations).

3.5 Error Analysis

As the above attempts to improve the recognition performance did not succeed, we looked at the errors more closely. First, we examined the phoneme sequence length statistics (i.e. when phoneme a occurs, how many following samples are also a). The averages were mostly accurate, except for the th phoneme, where recognised sequence length $l_{rec}(th) = 8.9$ was shorter than the validation data sequence length $l_{val}(th) = 10.6$, for oy (13.2/15.2) and for uw (1.4/10.2).
As this gives a hint that there may be a systematic error at work, we went one step further and analysed the transition matrix for the training and validation data T_{train} and compared it to the recognised transition matrix T_{rec}. This analysis showed that especially uw seems to be highly isolated, being reached by only few other states and transitioning to few other states, which is not the case in T_{train}.
This suggests that there may be a bias at work which may originate from the computed representation.

4 Improvement

Due to lack of time, we did not measure the improvement from HW1.