Modelling Discrete Data with Neural Networks

http://www.ru.why.cam.ac.uk/djw30/
djw30@maro.cam.ac.uk
Cavendish Laboratory, University of Cambridge
Interstellar Sciences Group
David Ward
POISONOUS, FLAT, SCALY, WHITE, BRUISED, PUNGENT, FREE, CLOSE.
EDIBLE, CONVEY, SCALY, YELLOW, BRUISED, ALMOND, FREE, CLOSE.
EDIBLE, CONVEY, SMOOTH, WHITE, BRUISED, ALMOND, FREE, CROWDED.
- Eg: descriptions of mushrooms (8 features shown).

Model independent samples of fixed length data.

**Task B**

- Would like to learn compositional rules such as punctuation and syntax.
- However, it only learns N-Grams.
- PPM (Prediction by Partial Match) is one of the best algorithms.

**Task A**

We want to model data streams - e.g. plain text, LaTeX code, C code.

**Introduction**
\[(\mathcal{H}, (u)_x|_{(u)}, \mathcal{I})_\mathcal{T} = (\mathcal{H}|_\mathcal{I})_\mathcal{D}\]

For example, in a bigram model, \(x\) could be a unary encoding of the previous character.

\[
(\mathcal{T}_u \{ (\gamma)^2 \})_f = (u)_x
\]

It is convenient to map the context, \(\mathcal{T}_u \{ (\gamma)^2 \}\), onto a vector \(x\).

\[
(\mathcal{T}_u \{ (\gamma)^2 \}|_{(u)^2})_\mathcal{T} = (\mathcal{D}_i)_\mathcal{D}
\]

The probability of the data can be written in terms of conditional probabilities

\[
\mathcal{T}_u \{ (u)^2 \} = \mathcal{D}
\]

The data is a sequence of symbols from an alphabet \(A\).

**Task A:** Modelling a stream of data
\[
\begin{bmatrix}
\ell_0 \mathbf{C} + \ell_1 \mathbf{C} \sum_{i=1}^{W} \tanh \end{bmatrix} \mathbf{y} = \mathbf{y}
\]

The matrix of weights \( \mathbf{C} \) maps input vectors \( \mathbf{x} \) to hidden units \( \mathbf{y} \).

\[
(\mathbf{W}';_{(u)} \mathbf{x})_{(u)} = ((u) \mathbf{x} \mid (u) \mathbf{z}) d
\]

such that

\[
(\mathbf{W}'; \mathbf{x}) \mathbf{y} = \mathbf{y}
\]
Motivation

\[
\sum_{i=1}^{f} \sum_{\varnothing} \sum_{V} \frac{\varnothing}{\varnothing} = \alpha
\]

\[
\alpha_{0} + \sum_{i=1}^{f} \sum_{\varnothing} \sum_{H} = \alpha
\]

Activation function: The matrix of weights \( \mathbf{W} \) maps the hidden units to outputs \( \mathbf{Y} \) through a softmax process.
\[
\frac{\left( N \mathcal{H} | \mathcal{D} \right) \mathcal{D}}{(N \mathcal{H} \mathcal{\varnothing}) \mathcal{D} (N \mathcal{H} \mathcal{\varnothing} \mathcal{\varnothing}) \mathcal{D}} = (N \mathcal{H} \mathcal{\varnothing} \mathcal{\varnothing} \mathcal{\varnothing}) \mathcal{D}
\]

Therefore, the posterior probability of the parameters is:

\[
[q \alpha - ] [q \alpha - ] \exp \left\{ \frac{1}{2} \sum Q - \right\} \sim (\alpha \mathcal{D}) \mathcal{D}
\]

Gamma prior on the hyperparameters (conjugate)

- In general, we will have multiple classes with different $\alpha$'s.

\[
\left\{ \alpha, C \right\} = \mathcal{W} \text{ weights by weights}
\]

The network is parameterized by weights $\mathcal{W}$.

\[
(N \mathcal{H} \mathcal{\varnothing} \mathcal{\varnothing} \mathcal{\varnothing}) \mathcal{D} \bigwedge_{(u) \mathcal{\varnothing}} = (N \mathcal{H} \mathcal{\varnothing} \mathcal{\varnothing} \mathcal{\varnothing}) \mathcal{D}
\]

The probability of the data is training.
Gibbs sampling of hyperparameters $\varphi$.

Hybrid Monte Carlo sampling of $w$.

To sample from

\[
\left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) \propto \left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) d \mathcal{I} \equiv \left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) d \mathcal{I}
\]

If we generate samples from $\{ \varphi, w \}$, then

\[
\left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) \propto \left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) d \mathcal{I} \propto \int \left( \mathcal{N} \mathcal{H} \mid \varphi, w \right) d \mathcal{I}
\]

Predictions are made by integration with respect to the parameters.

Monte Carlo Inference
I calculated the entropy of this model to be 2.066 bits / char.

Here is some data generated from this model:

accd 243 1 back 003 1 back 310 4 43 020 4 3 0 back 0 aace andc bbap
14 3 back acca 1 0 each 2 1 02 back 2 3 322 140 4 back 21 20 341 21 33

To make it more interesting, I made the distribution inside words slightly alphabetic.

{01234} => {s t a e d a c b}
State diagram. State s emits a space. States w emit symbols from {abeced}. States d!

A Toy Model
- Tested performance of the net on a large amount of unseen test text (JMB).
- Different length training texts.

The Hinton diagrams show the network architecture.

Input vector has weight 5 - bias plus unary encoding of last 4 characters.
3 hidden units
Toy Model

Log predictive probability / bits per char

Length of training data

PPM
net
uniform
H

Toy Model
We shall model the distribution of each instance \( Z \) in the following way

\[
\left( \left( Z \mid Z \right) p \right) \prod_{\text{features}} = \left( \left( Z \mid Z \right) p \right) \prod_{\text{features}}
\]

- flat, scaly, white, bruised, pungent, free, close.
- convex, scaly, yellow, bruised, almond, free, close.
- convex, smooth, white, bruised, almond, free, crowded.

Task B: Independent data of fixed length.
1186 test.
2000 training.
3186 strands: randomized and split.
60 features, each taking one of four values, ACGT.

DNA
4062 test
4062 training
8124 mushrooms: randomized and split into test and training sets
23 features, each taking in between 1 and 12 values

Mushroom

UCI Datasets
Future Work

- Neural net captures lots of information in the Mushroom data set

Task B

- Try English, C code, \LaTeX
- Results for the toy model are promising

Task A

Conclusion

- Application to text compression
- Slow - can we speed things up?
- Hyperparameter classes