# Review

Roxanne Yin

*Quantitative Finance & Risk Management, University of Michigan*

## 1 Unconstrained Online Handwriting Recognition with Recurrent Neutral Networks

**Abstract**

The article focus on unconstrained online handwriting recognition with a system which can directly transcribing raw online handwriting data without sophisticated preprocessing techniques. The system consists of an advanced recurrent neural network and performs very well on both raw and preprocessed data.

### Method

**Bidirectional LSTM:** Presenting the input data forwards and backwards to two separate hidden layers. use 'gate' units to store and access information over long periods of time.

**Connectionist Temporal Classification(CTC):** It trains the network to map directly from input sequences to the conditional probabilities of the possible labellings. The combined output sequence estimates the joint probability of all possible alignments of the input sequence with all possible labellings. Denote: set $L'^T = L \cup \{blank\}$ as paths, the conditional probability of a path $\pi \in L'^T$ is given by

$$p(\pi \mid \mathbf{x} = \prod_{t=1}^{T} y_{\pi_t}^t),$$

Then the paths are mapped onto labellings $\mathbf{l} \in \mathbf{L}^{\leq T}$ by an operator $\mathcal{B}$ to remove the repeated labels and blanks. Thus we have:

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x})$$

Now define a transformation of $\mathbf{L}^{\leq T}$ which is adding blanks between them denoted as $\mathbf{L}'^{\leq T}$. We have $|\mathbf{l}'| = 2|\mathbf{l}|+1$

forward variable $\alpha_t(s)$:

$$\alpha_t(s) = P(\pi_{1:t} : \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s/2}, \pi_t = \mathbf{l}'_s \mid \mathbf{x}) = \sum_{\pi:\mathcal{B}(\pi_{1:t})=\mathbf{l}_{1:s/2}} \prod_{t'=1}^{t} y_{\pi_{t'}}^{t'}$$

backward variables $\beta_t(s)$:

$$\beta_t(s) = P(\pi_{t+1:T} : \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s/2:|\mathbf{l}|}, \pi_t = \mathbf{l}'_s \mid \mathbf{x}) = \sum_{\pi:\mathcal{B}(\pi_{t:T})=\mathbf{l}_{s/2:|\mathbf{l}|}} \prod_{t'=t+1}^{T} y_{\pi_{t'}}^{t'}$$

Thus, we have

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{s=1}^{|\mathbf{l}'|} \alpha_t(s)\beta_t(s)$$

Objective function:

$$O^{CTC} = -\sum_{(\mathbf{x},\mathbf{z})\in S} \ln(p(\mathbf{z} \mid \mathbf{x}))$$

Using gradient descent for iteration:

$$\frac{\partial O^{CTC}}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{z} \mid \mathbf{x})} \sum_{s\in lab(\mathbf{z},k)} \alpha_t(s)\beta_t(s)$$

$$\mathbf{l}^* \approx \mathcal{B}(\arg\max_{\pi} p(\pi \mid \mathbf{x}))$$

**Integration with an External Grammar**

Using constraints with grammar G:

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p(\mathbf{l} \mid \mathbf{x}, G) = \arg\max_{\mathbf{l}} p(\mathbf{l} \mid \mathbf{x}) p(\mathbf{l} \mid G)$$

Then using *token passing algorithm* which allows us to find an approximate solution to equation above for a simple grammar.

## 2 Adaptive Market Making via Online Learning


**Abstract**

The article proposed a class of "spread-based" market making strategies and then design a master algorithm which obtains low regret relative to the best such strategy in hindsight.


**Framework**

*Regret* is the difference between the total value of the learner's algorithm and that of the best strategy in hindsight.

We assume events place on discrete points of stock's price. Denote *inventory* $H_t$ as amounts of stock holding at beginning period of time t; cash $C_t$ as money gain or loss in period time t; $V_{t+1}$ as the value of strategy at the end of period time t. The author also define **Market Order** and **Limit Order** in the article.

**Method**

**Spread-based Strategies:** This strategy consider a window size $b \in \{\delta, 2\delta, \cdots, B\}$, The window $[a_t, a_t + b]$ starting with $a_1 = p_1$, The when price p is out of this window, it will submits a buy or see order $\alpha$ at every price $p \in \Pi$, where $\Pi \coloneqq \{\delta, 2\delta, 2\delta, \cdots, M\}$. For this strategy, the profit is $kb\alpha$

**A low regret meta-algorithm:** The main idea is to weight all strategies $S(b)$ in each step. Here the author introduces two different methods to minize the regret which are **a low regret algorithm based on Mutiplicative Weights** and **a low regret algorithm based on Follow-The-Perturbed-Leader**.

The first algorithm takes parameters $\eta_t$, for $t = 1, 2, \cdots, T$. Initialize weights $\omega_1(b) = 1/N$ for every $b \in \mathcal{B}$. And it updates using the rule:

$$\omega_{t+1}(b) \coloneqq \omega_t(b) \exp\left(\eta_t(V_{t+1}(b) - V_t(b))\right)/Z_t$$

where $Z_t$ is the normalization constant to make $\omega_{t+1}$ a distribution.

The second algorithm requires a parameter $\eta$, and for every $b \in \mathcal{B}$, $p(b)$ be a sample from the exponential distribution with mean $1/\eta$. Then the distribution $\omega_t$ is set to be as:

$$\omega_t(b) = \Pr_p[V_t(b) + p(b) \geq V_t(b') + p(b') \quad \forall b' \in \mathcal{B}]$$