Learning When to Stop Thinking and Do Something!

Barnabás Póczos
Yasin Abbasi-Yadkori
Csaba Szepesvári
Russell Greiner
Nathan Sturtevant

Department of Computing Science, University of Alberta, Edmonton, CANADA

ICML 2009, Montreal, Canada
Contents

- Anytime algorithms
- Formal definitions
- Policy gradient approach
  - The Quality of the Estimated Gradient
  - Stopping Rule for Preventing Slow Convergence Near Optima
- Cross Entropy method
- Toy problem (mail sorting)
- Face detection
**Motivation**

- **Anytime algorithms**

  \[ t_1 < t_2 \implies r_{t_1} < r_{t_2} \]

- **Examples** (Important to be both accurate and efficient)
  - Mail sorter
  - Licence plate recognition on highway
  - Detecting possible terrorists on airports
Formal Definition of the Problem

- $X_1, X_2, \ldots$ iid sequence of instances, e.g. series of envelopes
- On $X_t$ we start a thinking process of max $K$ stages $Y_{t,1}, \ldots, Y_{t,K}$
- $Y_{tk} \in \mathcal{Y}_k$: information, (features) about $X_t$ at stage $k$ of thinking.
- $[q_k(0|Y_{t,k}), q_k(1|Y_{t,k})]$, $(k = 1, 2, \ldots, K)$, stopping policy.

<table>
<thead>
<tr>
<th>Observation:</th>
<th>$Y_{t,1}$</th>
<th>$Y_{t,2}$</th>
<th>$Y_{t,3}$</th>
<th>$\ldots$</th>
<th>$Y_{t,K}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1(0</td>
<td>Y_{t,1})$</td>
<td>$\tau_{t,1}$</td>
<td>$q_2(0</td>
<td>Y_{t,2})$</td>
<td>$\tau_{t,2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reward:</th>
<th>$r(X_t, \mu_1(Y_{t,1}))$</th>
<th>$r(X_t, \mu_2(Y_{t,2}))$</th>
<th>$r(X_t, \mu_K(Y_{t,K}))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action/Decision:</td>
<td>$\mu_1(Y_{t,1})$</td>
<td>$\mu_2(Y_{t,2})$</td>
<td>$\mu_K(Y_{t,K})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time:</th>
<th>$Y_{t,1}$</th>
<th>$Y_{t,2}$</th>
<th>$Y_{t,3}$</th>
<th>$\ldots$</th>
<th>$Y_{t,K}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_{t,1}$</td>
<td>$\tau_{t,2}$</td>
<td>$\tau_{t,K}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Formal Definition of the Problem

- $L_t$: stage when we quit. Total thinking time on $X_t$: $T_t = \sum_{k=1}^{L_t} \tau_{t,k}$
- $A_t = \mu_{L_t}(Y_{t,L_t})$ is the action taken on instance $X_t$

Ultimate goal:
$q = (q_1, \ldots, q_K)$

reward: $r(X_t, \mu_1(Y_{t,1}))$ $r(X_t, \mu_2(Y_{t,2}))$ $r(X_t, \mu_K(Y_{t,K}))$

action / decision: $\mu_1(Y_{t,1})$ $\mu_2(Y_{t,2})$ $\mu_K(Y_{t,K})$

observation: $Y_{t,1}$ $Y_{t,2}$ $Y_{t,3}$ $\ldots$ $Y_{t,K}$

time: $\tau_{t,1}$ $\tau_{t,2}$ $\tau_{t,K}$

$q_K(1|Y_{t,K}) = 1$
Learning Stopping Policies

\[ \rho^q = \mathbb{E} \left[ \lim_{t \to \infty} \frac{\sum_{s=1}^{t} r(X_s, A_s)}{\sum_{s=1}^{t} T_s} \right] \]

\{X_t\} is iid \Rightarrow \{R_s = r(X_s, A_s)\}, \{T_s\} are iid, too.

\[ \frac{\sum_{s=1}^{t} r(X_s, A_s)}{\sum_{s=1}^{t} T_s} = \frac{1}{t} \frac{\sum_{s=1}^{t} r(X_s, A_s)}{\sum_{s=1}^{t} T_s} \Rightarrow \rho^q = \frac{\mathbb{E}[r(X_1, A_1)]}{\mathbb{E}[T_1]} \]

The gradient of the policy can be calculated.

\[ \begin{aligned}
&\mathbb{E}[r_1] = \mathbb{E}[r(X_1, A_1)] \\
&\Delta \mathbb{E}[r_1] = \frac{\partial}{\partial \theta} \mathbb{E}[r(X_1, A_1)] \\
&\Delta \mathbb{E}[T_1] = \frac{\partial}{\partial \theta} \mathbb{E}[T_1] \\
\end{aligned} \]

\[ \frac{\partial}{\partial \theta} \rho^q = \frac{\partial \mathbb{E}[r_1]}{\partial \theta \mathbb{E}[T_1]} = \frac{\mathbb{E}[T_1] \Delta \mathbb{E}[r_1] - \mathbb{E}[r_1] \Delta \mathbb{E}[T_1]}{\mathbb{E}[T_1]^2} \]
Direct Gradient Ascent (DGA)

$$\frac{\partial}{\partial \theta} \log q = \frac{\partial}{\partial \theta} \frac{E[r(X_1, A_1)]}{E[T_1]} = \frac{E[T_1] \Delta E[r(X_1, A_1)] - E[r(X_1, A_1)] \Delta E[T_1]}{E[T_1]^2}$$

Approximation from $n$ instances:

$$E[T_1] \approx \frac{1}{n} \sum_{t=1}^{n} \sum_{k=1}^{L_t} \tau_k, \quad E[r(X_1, A_1)] \approx \frac{1}{n} \sum_{t=1}^{n} R_t$$

We need $\Delta E[r(X_1, A_1)]$ and $\Delta E[T_1]$.

They can be approximated using the **REINFORCE** algorithm.
Direct Gradient Ascent (DGA)

- \( Q_{tk} \sim q(\cdot | Y_{tk}) \)
- \( L_t: \) time when we quit.
  \[
  Q_{tk} = \begin{cases} 
  0, & \text{if } k \leq L_t \\
  1, & \text{otherwise.}
  \end{cases}
  \]
- \( \tilde{Y}_t = [X_t, Y_{t1}, \tilde{Q}_{t1}, Y_{t2}, \tilde{Q}_{t2}, \ldots, Y_{tK}, \tilde{Q}_{tK}] \),
- \( A_t = \mu_{L_t}(Y_{tL_t}) \) is the action taken on instance \( X_t \), \( r(X_t, A_t) = r(\tilde{Y}_t) \)

\[
\frac{\partial}{\partial \theta} \mathbb{E}[r(X_t, A_t)] = \frac{\partial}{\partial \theta} \int r(\tilde{y}) f_\theta(\tilde{y}) d\tilde{y} = \int r(\tilde{y}) \frac{\partial}{\partial \theta} (\ln f_\theta(\tilde{y})) f_\theta(\tilde{y}) d\tilde{y} = \mathbb{E}[r(\tilde{Y}_t) \frac{\partial}{\partial \theta} \ln f_\theta(\tilde{Y}_t)].
\]
Direct Gradient Ascent (DGA)

- Markov property along the trajectory: \( \tilde{y} = (y_0, y_1, \tilde{q}_1, \cdots, y_{K-1}, \tilde{q}_{K-1}, y_K) \)

\[
q_1(0|Y_{t,1}) \quad q_2(0|Y_{t,2}) \quad q_2(0|Y_{t,3}) \quad \cdots \quad q_K(1|Y_{t,K}) = 1
\]

\[
Y_{t,1} \quad \quad Y_{t,2} \quad \quad Y_{t,3} \quad \quad \cdots \quad \quad Y_{t,K}
\]

- \( \ell(\tilde{y}) = \min \{ k > 0 : \tilde{q}_k = 1 \} \) is the stage in \( \tilde{y} \) when \( q_\theta \) quits

\[
\ln f_\theta(\tilde{y}) = \ln f_0(y_0) + \sum_{k=1}^{\ell(\tilde{y})} \ln p_k(y_k|y_{k-1}) + \sum_{k=1}^{\ell(\tilde{y})-1} \ln q^\theta(0|y_k) + \ln q^\theta(1|y_{\ell(\tilde{y})})
\]

\[
\frac{\partial}{\partial \theta} \ln f_\theta(\tilde{Y}_t) = \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t})
\]
Direct Gradient Ascent (DGA)

$$\frac{\partial}{\partial \theta} \ln f_{\theta}(\tilde{Y}_t) = \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t})$$

$$\frac{\partial}{\partial \theta} \mathbb{E}[r(X_1, A_1)] \sim \bar{\Delta}_r \triangleq \frac{1}{n} \sum_{t=1}^{n} r(\tilde{Y}_t) \left[ \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t}) \right]$$

$$\frac{\partial}{\partial \theta} \mathbb{E}[T_1] \sim \bar{\Delta}_T \triangleq \frac{1}{n} \sum_{t=1}^{n} T_t \left[ \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t}) \right]$$
Direct Gradient Ascent (DGA)

$$\frac{\partial}{\partial \theta} \rho^q = \frac{\partial}{\partial \theta} \frac{\mathbb{E}[r(X_1, A_1)]}{\mathbb{E}[T_1]} = \frac{\mathbb{E}[T_1] \Delta \mathbb{E}[r(X_1, A_1)] - \mathbb{E}[r(X_1, A_1)] \Delta \mathbb{E}[T_1]}{\mathbb{E}[T_1]^2}$$

$$\frac{\partial}{\partial \theta} \mathbb{E}[r(X_1, A_1)] \sim \widehat{\Delta}_r = \frac{1}{n} \sum_{t=1}^{n} r(\tilde{Y}_t) \left[ \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t}) \right]$$

$$\frac{\partial}{\partial \theta} \mathbb{E}[T_1] \sim \widehat{\Delta}_T = \frac{1}{n} \sum_{t=1}^{n} T_t \left[ \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t}) \right]$$

Putting the pieces together:

$$\hat{G}_n = \frac{1}{n} \sum_{t=1}^{n} \left( \frac{r(\tilde{Y}_t)}{\widehat{T}} - \frac{\widehat{r} T_t}{\widehat{T}^2} \right) \times \left( \sum_{k=1}^{L_t-1} \frac{\partial}{\partial \theta} \ln q^\theta(0|Y_{tk}) + \frac{\partial}{\partial \theta} \ln q^\theta(1|Y_{tL_t}) \right)$$

where $\widehat{T} = \frac{1}{n} \sum_{t=1}^{n} \sum_{k=0}^{L_t} \tau k$ is the empirical average thinking time, and $\widehat{r} = \frac{1}{n} \sum_{t=1}^{n} r(\tilde{Y}_t)$ is the empirical average reward.
The Quality of the Estimated Gradient

\[ \hat{G}_n = \frac{\hat{\Delta}_r}{\hat{T}} - \frac{\hat{\Delta}_T}{\hat{T}}, \quad \hat{G} = \frac{\Delta \mathbb{E}[r_1]}{\mathbb{E}[T_1]} - \frac{\mathbb{E}[r_1]}{\mathbb{E}[T_1]} \frac{\Delta \mathbb{E}[T_1]}{\mathbb{E}[T_1]} . \]

**Hoeffding’s inequality:**

\[ \forall \delta \in [0, 1], \ \forall n, \ \exists \varepsilon_T, \varepsilon_r, \varepsilon_{\Delta_T}, \varepsilon_{\Delta_r} = O(\sqrt{\log(1/\delta)/n}) \text{ s.t.:} \]

\[ \mathbb{P}(|E[T_1] - \hat{T}| \leq \varepsilon_T) \geq 1 - \delta \]
\[ \mathbb{P}(|E[r_1] - \hat{r}| \leq \varepsilon_r) \geq 1 - \delta \]
\[ \mathbb{P}(|\Delta E[T_1] - \hat{\Delta}_T| \leq \varepsilon_{\Delta_T}) \geq 1 - \delta \]
\[ \mathbb{P}(|\Delta E[r_1] - \hat{\Delta}_r| \leq \varepsilon_{\Delta_r}) \geq 1 - \delta \]

\[ \|G - \hat{G}_n\| \leq \left\| \left( \frac{\Delta \mathbb{E}[r_1]}{\mathbb{E}[T_1]} - \frac{\hat{r}}{\hat{T}} \right) \frac{\Delta \mathbb{E}[T_1]}{\mathbb{E}[T_1]} \right\| + \left\| \frac{\hat{\Delta}_r}{\hat{T}} - \frac{\Delta \mathbb{E}[r_1]}{\mathbb{E}[T_1]} \right\| + \left\| \frac{\hat{r}}{\hat{T}} \left( \frac{\Delta \mathbb{E}[T_1]}{\mathbb{E}[T_1]} \right) \right\| \]
The Quality of the Estimated Gradient

**Proposition:**

\[ n \geq 2 \log\left(\frac{1}{\delta}\right) / \tau_0^2 \quad \Rightarrow \quad \mathbb{P}\left\{ \| G - \hat{G}_n \| \leq c_1 \frac{\sqrt{\log(4/\delta)}}{n} + c_2 \frac{\log(4/\delta)}{n} \right\} \geq 1 - \delta, \]

where

- \( \tau_0 \) is an almost sure lower bound on \( T_1 \),
- \( c_1, c_2 \) are constants that depend only on the range of the rewards, thinking times and their gradients.
Stopping Rule for Preventing Slow Convergence Near Optima

We know \[ \| G - \hat{G}_n \| \leq c(\delta, n) \leq \frac{1}{2} \| \hat{G}_n \| - \frac{1}{2} c(\delta, n) \leq \frac{1}{2} \| G \| \text{ (w.p. } 1 - \delta) \]

If \( n \) is large enough

\[ \| G - \hat{G}_n \| = O(\varepsilon) \text{ is not enough for gradient descent.} \]

We need \( \hat{G}_n^T G > 0 \), too. (e.g. \( \| G - \hat{G}_n \| \leq \frac{1}{2} \| G \| \) is enough)

\[ c(\delta, n) \leq \frac{1}{2} \| \hat{G}_n \| - \frac{1}{2} c(\delta, n) \quad (**) \]

Theorem:

Fix \( 0 < \delta < 1 \) and let \( n = n(\delta) \) be the first (random) time when (\( \ast \)) and (\( \ast \ast \)) hold.

Then \( \hat{G}_n^T G > 0 \) with probability \( \geq 1 - \delta \).
**Toy Problem: Sort envelopes**

- $h_t \sim U[0, 1]$ : difficulty of $X_t$
  - $h_t = 0$: difficult, $h_t = 1$: easy

- $p_{t,k} = \min\{h_t + 0.1 \times k, \ 1\}$, $k = 1, 2, \ldots, 5$
  - probability that this stage can classify $X_t$ correctly

- $y_{t,k} = (p_{t,k}, k)$, $k = 1, 2, \ldots, 5$ feature values

- $\phi(\cdot, \cdot) \in \{0, 1\}^{25}$ feature function for stopping policy:
  $$q^\theta(0|p_{t,k}, k) = g(\theta^T \phi(p_{t,k}, k))$$, where $g(x) = 1/(1 + e^{-x})$

- 1,000 random parameters ($\theta$)

- $n = 10$ random trajectories for each $\theta$

- Gradient descent for 1,000 time steps

- Before Gradient vs After Gradient
Toy Problem: sort envelopes

The performance histogram of a set of parameters before (BG) and after (AG) applying the DGA method
Cross Entropy Method (CE-DGA)

25 CE iterations, 100 samples, 10 elite samples, \( \lambda = 0.9 \)

\[ \theta \sim \mathcal{N}(\theta_0, \Theta_0) \]

Elite samples \( \Rightarrow (\bar{\theta}_0, \bar{\Theta}_0) \)

\[ \theta_1 = \lambda \theta_0 + (1 - \lambda) \bar{\theta}_0, \]

\[ \Theta_1 = \lambda \Theta_0 + (1 - \lambda) \bar{\Theta}_0 \]

\[ \theta \sim \mathcal{N}(\theta_1, \Theta_1) \]
Face Detection
For a given subwindow

Stage 1, $y_1 > \alpha_1$
- False

Stage 2, $y_2 > \alpha_2$
- False

Stage 3, $y_3 > \alpha_3$
- False

• **22 Stage classifier** (Lienhart et al. (2003))
• **Higher level classifiers perform better**
  - ... but have higher complexity

• **Training set:**
  - 4916 positive (faces), 7872 negative (nonfaces) of size 24x24

\[ \alpha \approx 0.5\%: \text{alarm for 99.5\% of faces} \]
\[ \beta \approx 30\%: 30\% \text{ of nonfaces are rejected} \]
Face Detection, Proposed Method

For a given subwindow

Stage 1

Stage 2

Stage 3

We are tuning $\alpha_k$ and $\beta_k$ on each stage

$$Y_{tk} < \alpha_k \implies \text{(non-face)}$$

$$\alpha_k < Y_{tk} < \beta_k \implies \text{(next-stage)}$$

$$\beta_k < Y_{tk} \implies \text{(face)}$$

$$q_k(\text{continue}|Y_{tk}) = \frac{1}{1 + \exp(-c(Y_{tk} - \alpha_k))} \times \frac{1}{1 + \exp(-c(\beta_k - Y_{tk}))}$$

$$R = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}$$
### Face Detection, Results

<table>
<thead>
<tr>
<th></th>
<th>VJ</th>
<th>Random</th>
<th>CE-DGA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E[R]</strong></td>
<td>96.95</td>
<td>76.80</td>
<td>97.70</td>
</tr>
<tr>
<td><strong>E[STAGE]</strong></td>
<td>13.17</td>
<td>1.25</td>
<td>6.1</td>
</tr>
<tr>
<td><strong>TPR</strong></td>
<td>96.70%</td>
<td>99.20%</td>
<td>97.00%</td>
</tr>
<tr>
<td><strong>FPR</strong></td>
<td>2.8%</td>
<td>45.60%</td>
<td>1.6%</td>
</tr>
</tbody>
</table>

*Expected reward, Expected stage numbers, True Positive Rate (TPR), False Positive Rate (FPR), for the Viola-Jones parameters (VJ), random parameters, and our method, respectively.*
Face Detection, ROC domain

Receiver Operating Characteristic (ROC)

- True positive
- False positive

- random
- VJ
- CE-DGA
Face Detection, Results

![Graph showing performance metrics over iterations](image)

- **CE-DGA**
- **CE**
Future Plans

- Better decision graphs for face detection?
  - What is the optimal structure?
  - Learning good features! as well, not only the thresholding constants in the decision functions, when both accuracy and efficiency matters.

- Other applications
  - Games
    - Hearts, preliminary results are promising
  - Path finding
Future Plans

- Empirical Bernstein bounds instead of Hoeffding?
- Our gradient estimation is biased…
  - Only the terms in it are unbiased
  - …but they are correlated
  - Independent estimators for the terms?
- Other gradient estimators?
  - Rao-Blackwellized Gradients
Conclusions, Contributions

- We investigated when to stop thinking about specific tasks when we consider both
  - the quality of our solution
  - and the cost of thinking

- Proposed an algorithm for this problem and showed its applicability on a face detection task.

- Hope to see you at our poster!

Email: poczos@cs.ualberta.ca

Thanks for your attention! 😊
The trade-off between time and work is particularly important in game competitions. Typically, a program will be given a fixed amount of time to play a game or a set of moves, but it is up to the author of the program to decide how to distribute the time between moves.

We chose to test the ideas in this paper in the game of Hearts using a UCT player. Hearts is normally played with perfect information, however most programs that play, do so by sampling the imperfect information and solving the perfect information samples. We report results on perfect information play here. UCT was used to make decisions in the game. A UCT sample consists of traversing from the root to a leaf in a UCT tree, and then performing a random sample from a leaf node to the end of the game. The program could choose to perform 100, 200, 300 or 400 UCT samples each turn. The work that this requires is determined by the position in the game tree. At the root of the game, 100 samples requires much more work than late in the game, as the tree being explored is deeper. Late in the game, most of the points have already been given out, so not only is less thinking required, thinking is also less useful.

The program has 15 features detailing information about the relative value of the moves being considered, the stage of the game, and points left in the game. Weights are assigned to the features, and the program decides to keep thinking about a particular move if the dot-product of the features and weights is greater than 0. Weights on the features are trained using cross-entropy. A fixed set of 50 games was used to evaluate a policy for the training, with the scoring function as the score/(work+k), where $k$ is a parameter which controls the preference between score and work. $k$ was tuned by hand to favor both equally.

After this training, the learned program was measured to perform 2000 node expansions on average per move. We then determined that a program which statically uses 325 samples per move performs the same number of node expansions on average. Because the number of samples is fixed, the amount of work is biased towards earlier in the game.

The static and learned policies were compared competitively in 100 games played to 100 points. As Hearts is played with four players, we played each game with all possible combinations of the two player types (static and learned) in each seat, to account for biases. There are 16 ways to place two types of players in a 4-player game, but two of those contain all of one or the other type, so these aren't used. This means that each game is played 14 times, for 1400 games. Over these games, the learned policy beat the static policy by a margin of 2.7 points (73.4 to 76.1; lower scores are better). While the margin is relatively small, it is statistically significant with 99% confidence.

Now that we better understand how to successfully apply this trade-off in a small game, we are interested in extending these results to games where more time is spent per move.
Hearts

- Hearts is a simple game to learn, but provides ample opportunities for strategy and daring. The objective is to earn as few points as possible. Each Heart card is worth one point, the Queen of Spades is 13 points, and other cards have no point value. In each hand, the highest card in the suit that was led wins the trick. When a player reaches 100 points, the player with the lowest point total wins.
true positive (TP), eqv. with hit
true negative (TN), eqv. with correct rejection
false positive (FP), eqv. with false alarm, Type I error
false negative (FN), eqv. with miss, Type II error
true positive rate (TPR), eqv. with hit rate, recall, sensitivity
  \( TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \)
false positive rate (FPR), eqv. with false alarm rate, fall-out
  \( FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \)
accuracy (ACC)
  \( ACC = \frac{(TP + TN)}{(P + N)} \)
specificity (SPC) or True Negative Rate
  \( SPC = \frac{TN}{N} = \frac{TN}{(FP + TN)} = 1 - FPR \)
positive predictive value (PPV), eqv. with precision
  \( PPV = \frac{TP}{(TP + FP)} \)
negative predictive value (NPV)
  \( NPV = \frac{TN}{(TN + FN)} \)
false discovery rate (FDR)
  \( FDR = \frac{FP}{(FP + TP)} \)
\[ f = (\ldots, f(-1), f(0), f(1), \ldots) \text{ (two-sided)} \]

\[ |f(k)| < \begin{cases} 
  Ar_1^k & k = 0, 1, \ldots \\
  Ar_2^k & k = -1, 2, \ldots 
\end{cases} \]

\[ 0 < r_1 < |z| < r_2 \]

\[ \Rightarrow F(z) \text{ is absolutely convergent.} \]