集合知プログラミング
第4章　検索とランキング

04T4027T　斉藤章久

はじめに

・これまで見てきた基準はすべてページの内容に基づいたものだった。

しかし、ページに対して提供する情報を考慮することによって、多くの場合結果を改善することができる。

インバウンドリンクの利用

- 一番簡単な利用方法は、それぞれのページのインバウンドリンクを数え上げ、リンクの合計をそのページの測定基準として利用する。

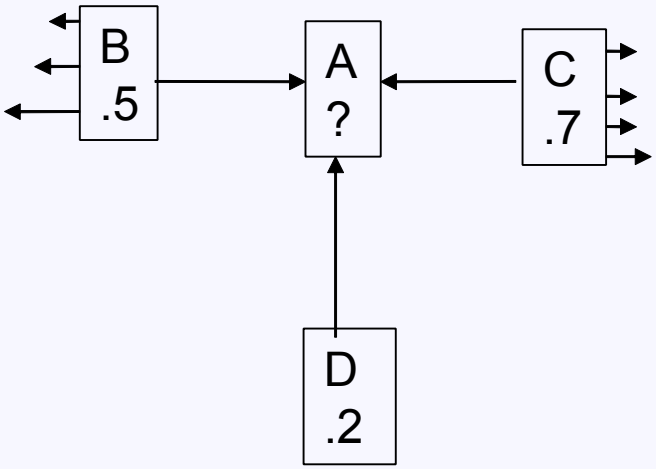
（例）

論文の価値を評価するとき、その論文を参照している他の論文の数で、その論文の価値を評価する。

PageRankアルゴリズム（1）

- pageRankアルゴリズムはGoogleの創設者たちによって発明された。
- このアルゴリズムはすべてのページにそのページがどの程度重要なのかというスコアを割り当てる。その重要度はそのページにリンクしている他のページの重要度の合計と、それらの他のページたちがそれぞれ持っているリンクの数から算出される

PageRankアルゴリズム（２）



例えば、B・C・DはAにリンクを張っていて、Ａ以外のリンクをBは3つ、Cは4つ、Dは0とする。

B・C・DのPageRankを0.5・0.7・0.2とすると、次のような式でAのPageRankはもとまる

PR(A)=0.15+0.85*(PB(B) / link(B)+PR(C) / link(C)+PR(D) / link(D))

PageRankアルゴリズム(3)

- （2）の計算はすべてのページにPageRankを持っていなければならない

- PageRankを事前に持っていない場合、初期値として任意の値をすべてのPageRankとして用いる。そして（2）の計算を何度か繰り返すと、そのページの本当のPageRankの値に近づいていく。

リンクのテキストを利用する

- そのページへのリンクに含まれている文字列を利用する
- ページ先へのコメントの情報の方が、リンクしているページそのものよりも情報が含まれていることが多い。
- 検索する単語がリンクに含まれているかどうか探す。もし、リンク先が検索結果のどれかにマッチすると、リンク元のPageRankがリンク先の最終的なスコアに付け加える。重要なページから検索後を含んだリンクが数多く貼られているページのスコアが高くなる。

クリックからの学習

ユーザによる結果検索のクリックを記録し、その記録をりようして検索結果のランキングを改良する方法を検討する。

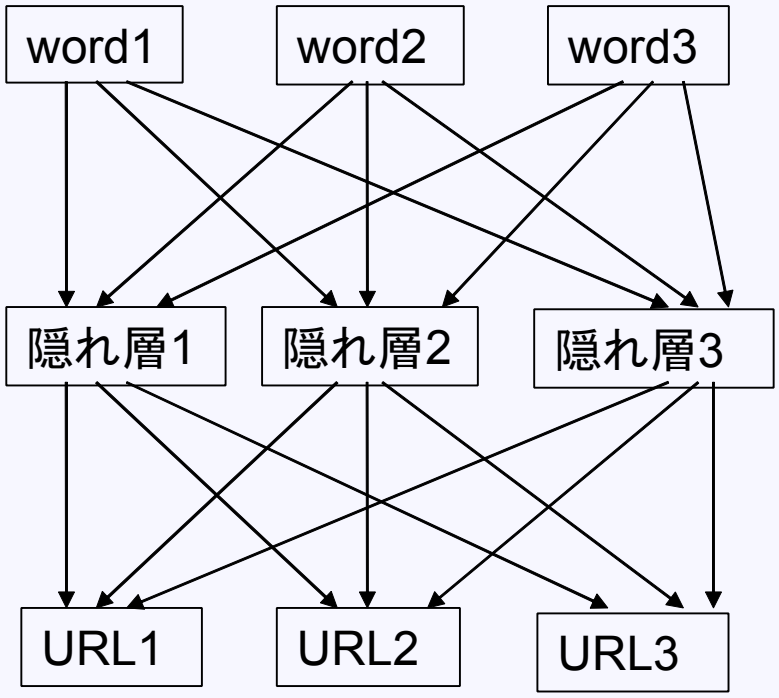
これを行うために、クエリ中の検索語、利用者に表示される検索結果、ユーザがどれをクリックしたかという情報を基にトレーニングを行う人工的なニューラルネットワークを作り上げていく。

クリックを追跡するネットワークの設計（1）

- この章で学ぶネットワークは多層パーセプトロンネットワークと呼ばれるものである。

  このタイプのネットワークは複数のニューロンの層から構成され、最初の層は入力を受け付け、最後の層が出力を行う。中層部は外部と交流することは決してないため、隠れ層と呼ばれ、入力の組み合わせに反応する。

クリックを追跡するネットワークの設計（2）



クエリ中の単語に該当する入力ノードの値を1にセットする。それらのノードから出力が行われ、隠れ層を発火しようと試みる。次に出力層のノードたちを発火しようと試みる。

それらの発火レベルがURL元のクエリ中の単語との関連の強さを決めるのに使われる。

データベースのセットアップ（１）

ニュートラルネットワークはユーザがクエリを実行するたびにトレーニングされなけらばならない。

↓

ネットワーク表現をデータベースに保存しておく必要がある。

必要なのは隠れ層のテーブルを一つと、コネクションのための二つのテーブルである。

データベースのセットアップ（2）

- getstrengthというメソッドを作る。これは、もしコネクションが存在しない場合にはデフォルト値を返す。単語から隠れ層へのリンクのデフォルト値は-0.2、隠れ層からURLへはデフォルト値として0を返す。

- setstrengthというメソッドを作る。これは、コネクションが存在するかどうか調べて、存在すれば新たな強度でコネクションを更新したり、コネクションを作成する。

フィードフォワード（１）

- フィードフォワードアルゴリズムとは、入力のリストを受け取ると、ネットワーク内を通貨させ、出力層のすべてのノードの出力を返すアルゴリズムである。

- 最初にそれぞれのノードが入力に対してどの程度反応するかを示す関数を選択する。ここではハイパボリックタンジェント関数を使う。

  x軸はノードへの入力の合計とし、y軸を出力とする。入力が0に近づくにつれ出力は急激に上昇を始める。入力が2になった時は出力はほとんど１であり、それ以上は上がらない。

フィードフォワード（２）

・クラスはデータベースに対し、ノードとコネクションについて問い合わせを行い、メモリ中に特定のクエリに関するネットワークを作り上げる必要がある。

最初のステップとして、特定のクエリに関する隠れ層のすべてのノードを探し出す関数を作る。

・データベースから引き出した現在のすべての重みで、ネットワークを構築するメソッドも必要である。

この関数はこのクラスの多くのインスタンス変数を設定する。

（単語のリスト、クエリノードとURL、すべてのノードの出力レベル、すべてのノード間のリンクの重みなど）

フィードフォワード（３）

- フィードフォワードアルゴリズムは隠れ層の中のすべてのノードをループし、入力層からの出力に対しリンクの強度を掛け合わせ、足し合わせる。

  それぞれのノードの出力は入力の合計にtanh関数を適用したものであり、出力層に渡される。

  ↓

  つまり、前の層からの出力に自身の強度を掛け合わせ、その値にtanh関数を適用して最終的な出力をつくり出すのである。

バックプロパゲーションによるトレーニング（1）

このネットワークは出力を出すが「よい結果」というものがどういうものかは教えられていないため、結果は役に立たない。

↓

改善するには、ネットワークが教えてもらった「正答」をより反映するためにノード間のリンクの重みを変更するアルゴリズムが必要である。

この改善を利用するアルゴリズムは、ネットワークの中の重みを調整しながら後ろに伝わって行くため、バックプロパゲーションと呼ばれるものを利用する。

バックプロパゲーションによるトレーニング（2）

- 人は出力層のそれぞれのノードの望ましい出力を常に知っている。この場合、もしユーザがクリックしたらその出力を1に近づけるべきであり、押されなかったら0に近づけるべきである。あるノードの出力を変更するには、そのノードへの入力の合計を変更する以外に方法はない。

- バックプロパゲーションのメソッドを動かす前に、すべてのノードの現在の出力をインスタンス変数に保存しておくためのfeedforwardを動かす必要がある。

バックプロパゲーションによるトレーニング（３）

【出力層のそれぞれのノードに対して次のステップを実行する】

（１）ノードの現在の出力とあるべき出力の差を計算する

（２）dtanh関数を使ってノードの入力の合計をどれくらい変更すべきか決める。

（３）入ってくるリンクすべての強度を、リンクの現在の強度と学習率に見合うよう変更する。

【隠れ層のそれぞれのノードに対して次のステップを実行する】

（１）ノードの出力を、それぞれの出力リンクの強度の合計に目標のノードをどの程度変更するべきかという値を、掛け合わせた値によって変更する。

（２）dtanh関数を使ってノードの入力の合計をどれくらい変更すべきか決める。

（３）すべての入力リンクの強度を、リンクの現在の強度と学習率に見合うよう変更する。

トレーニングのテスト

- 結果例でトレーニングすると、その結果以上の出力が増加する。
- このネットワークはどのURLがどのクエリに関連しているかを学んだだけでなく、特定のクエリ中のどの単語が重要なのかについても学習している。

検索エンジンとつなげる

- searcherクラスのqueryメソッドは結果を作り表示する過程で、URL IDのリストとword IDのリストを取得する。このメソッドにこれらを返させるようにする。
- 人工的なニューラルネットワークを作りあげるための最後のステップは、searcherクラスの中に検索結果を重み付けする新しいメソッドを作り上げることである。

- この章で開発した検索エンジンは、数百万におよぶ規模のページをインデックスする際には考える必要があるが、10万ページ程度であれば問題なく動作する。