簡単なコネクショニストモデル入門講座（第ゼロ回）

面倒な数学が判らなくてもシミュレーションはできる！！！

参加資格

- 自由にプログラムを起動、終了、ある場所の特定ができる(つまりの OS の基本操作が独力でできる)
- テクスト・エディタ(Emacs, vi(m), BBEdit, UltraEdit, Kate などなど) が使える
- プログラミングを(少しなら)勉強する気がある
- 正規表現(Regular Expressions)を学ぶ気力がある
- 人に(特にチューターに)尋ねる前に、とりあえず自分で考えてみることができる

このコースが目標とすること

- コネクショニスト・モデルの実際は難しくないことを理解する
  - 数式の理解は不可欠というわけではない
- 動作原理を数学的に理解していなくても，とりあえずシミュレータを使えるようになる
  - タスクを上手く定義する
  - 訓練データを上手く作成する
  - 上手い符号化を使う
- コネクショニスト・モデルで何ができて何ができないのかを理解する
- 認知言語学が本当にコネクショニスト・モデルと折り合うのか確かめる

使用教材

- 教科書: *Excercises in Rethinking Innatenss* (Plunkett and Elman 1996, MIT Press) から適当な節を抜粋
- シミュレータ: tlearn (http://crl.ucsd.edu/software で無料入手可能: Mac, Windows, Unix 版あり)
- シミュレータの関連ファイル一覧
  - *.cf ファイル(ネットワークの設定ファイル)
  - *.data ファイル(入力パターンを 1 行につき一個指定)
  - *.teach ファイル(標的パターンを 1 行につき一個指定)
  - *.err ファイル
  - *.data, *.teach は二つで一セット

なぜコネクショニスト・モデルなのか？

- 規則ベースのシステムに比べて有利な点
  - グレーディエンスのある、揺らぎのある振舞いをモデル化するのに適している
  - データに内在する曖昧性を許容する
  - データ内部のノイズを上手く除去する（ただし「過剰でないならば」という条件付き）
  - 経験から学習し、それから構造を創発させるシステム
  - 相転移的な挙動を示す

コネクショニスト・シミュレーションを自分でやってみる意義

- <言語の規則>論争を理解し，再解釈する
  - Rumelhart and McClelland 1986 (コネクショニスト) と Pinker and Prince 1986 (生成文法) の論争の争点を肌で理解する
- 言語の認知的モデルの動作原理を，実際にそれを自分で動作させてみて理解する
  - 例えば Langacker 1991 が「認知文法がコネクショニストモデルと互換だ」と主張する際，その妥当性の根拠が単に terminological なものでないか，自分の目で確認する

論文では論じられないコネクショニスト・モデルの基本

- 適切な課題のデザイン
  - 適切な(i.e.,うまくいく)課題のデザイン
  - 適切な(i.e.,うまくいく)訓練データの作成
  - 適切な(i.e.,うまくいく)符号化
  - 適切な(i.e.,うまくいく)アーキテクチャの選択
  - 適切な(i.e.,うまくいく)パラメタの設定
- 適切な結果の評価方法
  - 上手くいかなかった時に何がまずかったのかを洞察する方法
  - たいていのシミュレーションは始めからうまくいったりしない！

Artificial Neural Network (ANN) の基本アーキテクチャ

- 単純パーセプトロン(隠れユニットなしの一層)
- 多層パーセプトロン(隠れユニットありの二層以上)
- 回帰型(feedback あり) vs 非回帰型 (feedback なし=進行型 feedforward)
- モジュラー vs 非モジュラー
  - ただし、モジュラーネットは当コースでは扱わない
- そのほかにも分類はあるが，当面はこれだけで十分

とにかく、やってみよう！！

次回のための準備

- *Exercises in Rethinking Innateness*: Chapter 3 を読んで来る
- できれば、背景の説明である Chapters 1,2 も