心理学とプログラミング ： 数理の壁を越えて

著者	岡本 安晴
雑誌名	文化情報学
巻	6
号	1
ページ	38-46
発行年	2011-03-10
権利	同志社大学文化情報学会
URL	http://doi.org/10.14988/pa.2017.0000013114

Journal of Culture and Information Science, 6(1), 38-46.（March 2011）

講演記録

心理学とプログラミングー数理の壁を越えてー

日本女子大学人間社会学部心理学科教授
岡本　安晴

日本女子大学の岡本です。よろしくお願いします。「数理の壁を越えて」と書いていますが、心理学や社会系の場合には、数理モデルで表したときどうしても複雑になってしまって、解析的にもまともに解くということが不可能な場合が普通です。その場合に、数値計算というか、そういうプログラミングで様子を調べるという方法がとられているわけですが、それを一応、この「数理の壁を越えて」という言葉で表したしだいです。

それから、大学院生の方が来られているということですが、大学1年生ぐらいの人が中心になるのかなと思い込んでいましたので、一応そういう学部の1年生で、心理学もまだ専門に勉強なさっていない人を対象と考えて用意してきました。今ここで急に大学院生の内容に切り替えるということも難しいですので、そこのところはそのようなつもりで準備がされたのだと了解してください。

このタイトルで、三つ言葉を並べてありますが、心理学は、日本では人文系に分類しています。アメリカでは理系になるのですけれども、日本では、心理学は文学部や、あと教育学部、最近では工学部もけっこうやっています。それとこの数理の世界。数理というと代表的なのは理学部、あと工学部という理系の世界になるわけです。これを、ある意味で異質というか、人文系のものに数理を当てはめるということについては疑問を持つ人もかなりいると思うのですけれども、それでも実際にはどんどんと、この数理的な手法が人文系で取り入れられているというのは、もう皆さん方にとっては当たり前のことだと思います。

ただ、人文系に数理を持ち込んだときには、どうしても複雑にならざるを得ないので、この間はプログラミングという方法でつなぐことによって、数理を生かした人文系の研究が展開できるという、そういうことを表してみたわけですが、今回講演の機会を与えていただいたときに、まず、なぜ理学部から心理学に移ったのか、その経緯についての説明と、それから、心理学でプログラミングをどのように活用しているのか。たぶんこれがメインだと思うのですけれども。そして私が現在どういうことをしているか、そういうことを内容で用意するということでお話をいただきました。

この理学部から心理学という、ここのところを簡単に自己紹介させていただきますと、なぜ私が理学部へ行ったかというと、それは数学があったからというのが直接の理由です。なぜ数学かというと、数学をやりたくて数学に行ったのではないのですね。やりたかったのは、考えるということについて、簡単な言い方をすれば、正しく考えるとはどういうことかと。それは哲学の考え方ではなくて、もっと形式的な論理でやってみたいと。そのとき形式的というのは考えていませんでしたけれども、哲学的なやり方は合わないですね。もっと、より気持ちとして合う言い方をすれば、客観的な、基準のしっかりしたアプローチで考えるということを勉強してみたい、それはどうも数学らしいと、数学が一番近いらしいということで、それで数学を勉強しようと思いました。それは、『形式論理学』という本を高校のときに見たということもあります。形式論理学といっても、それは数学基礎論のほうの形式論理学の本だったのですけれども、そういうことで、ともかく理学部に入ったと。

理学部に入ったわけですが、いろいろなことを教養的科目として勉強させられるわけです。そのとき、私が一番楽しかったのは物理学でした。物理学の問題をいろいろ解くと、このときが一番楽しかった。でも、大学へ入った目的は、考えるという、論理について客観的に勉強したいと。京大の場合は、その領域の方というか、そういうゼミは理学部にはなかったと思うのですけれども、最初考えた目的に一番近いところというわけで、数

学のなかでも抽象代数に近いところを選びました。

やらなければいけないことと自分のしたいこととは、違う場合があります。私の場合は、やらなければいけないという観点からいけば、論理学を勉強するために大学へ入ったのですから数学ということになるわけですが、やはり現象を数理的にいろいろ見るということが好きなのですね。ただ、それだけならまだあまり問題はなかったと思うのですが、実際には物理にかわることもちょっと考えたことがありました。でも、今から40年ほど前、物理学の本を見ると、素粒子のややこしいことがたくさん書いてあったのですね。これを全部覚えないと物理学の勉強はできないのかと思うと、ちょっとうんざりしまして。また、素粒子の性質についての一覧表をつくって、それを眺めながらするというのも考えましたけれども、どうもそれは楽しそうではないという気がしました。

決定的だったのは、高校のときに臨床的な心理学にもけっこう興味があったものですから、大学へ入って、心理学の授業が取れるようになっていましたので心理学を取りましたが、その心理学は臨床とは全然関係のない心理学で、人も物理学と同じような手法で研究できるという、そういうことが紹介されていました。これは、大学へ入って初めて知った考え方であって、そのことがずっと尾を引いていました。いろいろなことがあって、結局、最後に社会へ出る前に、人を物理的に、あるいは工学的に研究するということも勉強してみたいと。それを親も許してくれましたので心理学へ移ったというしだいです。ですから心理学も、私の場合には、物理的な手法でアプローチするということが楽しいという状態です。

心理学のことを話す前に、数理的にアプローチしているわけですけれども、この数理ということについてちょっとまとめてみました。ここで一応、研究対象を科学的操作可能性と関連付ける表現形式というように書きました。例えばそれはニュートン力学の場合、これは言語的に表現すれば、木から離れたら落ちると、ただそれだけなのですね。別の言い方をすると、力というか、下から引っ張られるとどんどん落ちていくと。でも、これを数理表現としてこの運動方程式の表現を使うと、現在では、厳密には相対論の効果とか入れますけれども、基本的にはこのような数理モデルを設定することによっていろいろなコントロールが可能になっているわけです。人工衛星が飛ぶとか、電車が動くとか、そういうことが可能になる。つまり言語表現から数理的表現に進むというか、数理的表現を使うことによって、その法則の利用度がうんと違ってくるということだと思います。

それと最近言われていることは、データの可視化、見えるようにして、見えるかたちでビジュアルにデータの情報を表すということ。今、アメリカの心理学ではグラフで結果を表示しなさいということが非常に強調されていますし、日本でもいろいろなことを絵で説明するということが重要視されています。その絵で表現するときにプログラミングを使うわけですけれども、この絵で表現するということは、座標軸を設定して、そしてグラフを書くというのが一つの方法なわけです。すなわちこのように数理的に表現することによって、関係のビジュアルな表現が可能になる。そういう現実の利便性もあると、そのように思います。

そのビジュアル化という利便性もあるわけですが、数理的に表現するとより深いことが分かるというのは、心理学で見てみますと、例えば、これは横軸に体重、縦軸に知っている漢字の数を取ったときに、1年生で取るとこれぐらいの分布、6年生について取るとこれくらいの分布ということであると。6年生分全部まとめてしまいますと、体重が増えるほど知っている数が多いということになるわけです。すなわち体重と知っている漢字の数とは、正の相関がある。では、子どもにたくさん漢字を覚えさせるには体重を増やせばいいかということになりかねないわけですね。こんなばかばかしいことはないと、すぐこの場合には気がつくわけですが、ほかのもっと微妙な問題のときには、こういう論法がけっこうまかり通っているわけです。

これは、仮に相関係数が、体重と漢字の数で0.56あったというときに、この学年という統計変数というか、第三の変数を持ってきて、そしてこの影響を、統計モデルを使って除いてやると、学年の影響を除けば、このなかのそれぞれ学年内での関係になりますから相関がなくなる。つまり、体重と知っている漢字の数との関係は、学年という第三の要因から生じた疑似相関だということになるわけです。これは数理モデルで関係をきちっと表したことによって初めて出てくる事柄なわけです。

もう一つ例を言いますと、工場での作業のエラーを見るときに、知能の高い人ほどエラーが少ない

かというのを調べてみたら関係がなかった。知性のある人は賢いから、ミスが少ないだろうと思ってデータを取ってみたら関係がありませんということが出てしまったと。そうかということになるわけですが、これは第三の変数を設定した統計の解析のための数理モデルを設定します。すなわち飽きっぽさ。知能の高い人は、すぐ一つのことに対して飽きてしまうのではないかと。この飽きるということがエラーに関係するのではないかということで、その影響を調べる統計モデルを立てて分析してみると、その知能から飽きっぽさに対する影響が0.69という係数、重みで影響があると。飽きるとエラーが0.71というかたちでエラーが生じやすくなる。

それから、知能の高い人はエラーを起こしにくいですね。知能とは何かという問題もありますが、知能が高い人はエラーを起こしにくい。それがマイナス0.50である。これが正しい全体の関係であるときに、ここの飽きっぽさというのを抜いて、この二つだけの関係を見ると、あたかも無関係になる。このような関係は、こういう数理モデルを設定して、この赤い線で表した関係が浮かび上がってくるということで、心理学においてもこの数理モデルを使わないと、真の関係が浮かび上がってこないということが分かるというわけです。

その心理学で数理モデルが必要だということですが、心理学とはどういうものであるかというのを少し説明してみたいというので用意したのがこれです。大学1年生ぐらいの人がいると、心理学のことも説明してから話を進めたほうがいいかなと思って用意しました。せっかく用意したので、見ていてください。

文化の担い手である人について科学的に研究する。科学的というのは情報科学などの活用ということで考えてみましたが、人について科学的に研究するという観点で心理学をとらえたとき、心理学は、社会心理学、認知心理学、言語心理学とか何とか心理学と、幾つかありますが、心理学をこのような関係で見たときに、では心理学というのは、だいたいいつから、どういうかたちで始まったのかということですが、これは実験心理学を中心にして考えております。

実験心理学がいつ始まったかというと、フェヒナーという人が、これはドイツ語ですけれども、英訳だと、"Elements of Psychophysics"になりますが、この書物を著したときを初めとする考えが一つとしてあります。実験データだけを取り上げるならいろいろあるわけです。でも、データを集めただけでは科学ではないという考えでいきますと、データと理論との一つのまとまりとして提示したというのはこのフェヒナーに始まると、そのように説明する本がけっこうあります。もう一つ、ヴントという人が、ライプチヒ大学に心理学の実験室をつくったと。このときを実験心理学の始まりとして説明する場合もあります。

実験心理学という範囲でいくとそういうことになるわけですが、もう少し広く考えると、有名なフロイトですね、精神分析学のフロイトは、こういう19世紀から20世紀の変わりめに活躍した方です。それから発達心理学で有名なビネーとかピアジェとかいう人はこの年代です。

普通、心理学だとここまでですが、心理学に関係するものとして、色についての研究があります。色がどう見えるかというのは、これは心理学における一つの大きなテーマですが、その色についての研究ということであれば、ニュートンが、光ですね、白色というか、普通の白い光、これをスペクトルに分解したというのがある。それから、その色の見え方ですね、色には三つの基になる色がある。これは色を見る細胞である錐体という感覚細胞が網膜にありますが、人間の場合は、多くの人はそれを3種類持っているということに対応しているのだと説明されていますけれども、それがこのあたりですね。ニュートンの100年ほど後になります。それからこの同じ時期に、このような三色説とは別に、赤と緑、これは同時に色のなかに感じることができないとか、そういう意味で反対色という、そういう考え方を出したヘーリングの反対色説というのがこの年代です。それから記憶の実験ですね、それはエビングハウスという人が記憶ということで実験をしました。被験者は自分です。自分自身を使って実験したわけです。

このように見てくると、19世紀の終わりごろに実験心理学というのは始まっている。そして今、あなた方が認知心理学とかというので聞いているそういう心理学にきているということになると思います。

こういうものを踏まえたうえで、心理学におけるプログラミングがどういう位置付けか、有効性があるかというわけですが、人文系というのは、これは複雑システムであると、いろいろな要因が効いてくるので複雑システムととらえたときに、

それは、複雑であるがゆえに数理的解析が困難であるということになります。その困難なものに対してどうするかというところで、最初のタイトルに出しました、この困難さを越えるためにプログラミングというものが使えると。

この数理の壁を越えるためのプログラミングのほかに、心理学では実験のときに、実験を自動化するために、実験制御のプログラミングをおこなうということ、これは昔のマイコンと呼ばれた時代からそれはおこなわれています。マイコンの前のリレー計算機という、電磁リレーで論理回路を組むというのが昔はあったわけです。どれくらい昔かというと、50年ぐらい昔ですけれども。そのころから実験の自動化ということで、論理回路と呼ばれているもので自動化するということはおこなわれていました。それがマイコンとか、そういうものが出てきましたので、プログラミングでソフト的に対処できるようになったということになります。

このプログラミングがどう使われているかということですが、結論から言うと、不可能であったことがプログラミングによって可能になったものがかなりあります。例えばどういうことかと言いますと、多変量データといって、一つのケースから何百というデータを得るのが人文社会系ではけっこうあるわけですが、それを分析するときには因子分析とかそういうことをするときには固有値を求めるということになるわけです。固有値というのは、高次の多項式の解です。三次か四次ぐらいまでなら公式がありますけれども、百次とか二百次とか一万次とかになってくると、これは解析的に解こうなんて誰も思わないですね。回帰分析のときには逆行列で計算するわけです。何百という独立変数を使う場合には、何百次の行列の逆行列を求めるということになりますので、これも手計算では難しい。でも、因子分析なんかのときに、どうしても因子分析したいということで、そろばんのアルバイトの人を何十人もお願いして、それは科研費で予算を取るわけですが、1カ月も2カ月もかけて解を得たという話はあります。今なら1秒もかからないような計算を1カ月も2カ月もかけて。それは必要だからやったわけですけれども。

実用的なレベルで数学的に解くのが不可能なものが、今では数値計算法で簡単に出ると。今日どこかで見たのでは、スーパーコンピュータに匹敵する能力のパソコンが用意されたと言いますが、グラフィックボードに搭載されているGPUが並列計算、何百というのが同時にできるので、それを使うとかつてのスーパーコンピュータ並みの計算速度になるということですけれども、今はそういう時代です。アプリケーションで利用するという方法もありますけれども、ともかくそういうことが可能になったと。

それから、これはデータ分析ですが、人文系の場合、複雑なモデル、特にニューラルネットワーク、現在よく盛んに研究されているものはニューラルネットワークですが、これを数学的に解析するというのは、エントロピーが安定するとかどうとか、一般論はできますが、個別の場合においてどのような振る舞いを示すかと。もうこれはシミュレーションの世界になるわけですね。そのようなシミュレーションで使えると。それからこの実験の自動化、インタラクティブな実験というのは、被験者の反応を見ながら、その被験者の反応を分析しつつ、その分析結果に基づいて次の実験のステージへ進むという、そのインタラクティブな実験、それが可能になってきたと。そういうのでプログラミングが使われるということになります。

一般論としてはそういうことですが、では話を心理学に限ってみますと、まず、今はコンピュータの時代ですからそれを活用する。例えば心理学実験の自動化ということがありますし、それからデータから情報を取り出す、データサイエンスという言葉が使われていますが、それも心理学でやります。それから、これはかつてよくおこなわれたことですが、コンピュータになぞらえて人間の知的活動を説明するということもあります。それからシミュレーションですね。ニューラルネットワークとか、モデル自体は心理学独自の、上の、比喩ではないのですけれども、ただそれは数理的解析が難しいのでシミュレーションでやるというわけです。

このときに、プログラミングと書きましたが、既製のアプリケーションを使うということももちろんあるわけです。私も簡単なときには既製のアプリケーションでけっこうやっていますが、そういう、すでに用意されているものがあるのに、なぜプログラミングということが意味があるのかということで、ちょっとこの対比をおこなってみました。

アプリケーションは、一応そのアプリケーショ

ンが想定している決まりきったことを処理するというのにはいいわけです。ただ、決まりきったことというのは、いろいろなことに対応できるためには、それに対応したたくさんの機能をそのアプリケーションは備えなければいけない。そうすると複雑になるわけですね。マニュアルは最近、パンフレットというか書籍の形で収まりきらないぐらいの量になってきたので、PDFファイルとか、電子化されたもので同梱されているのが最近多いですよね。数年前まではちゃんとマニュアルがついていたのですが、最近のアプリケーションはもうマニュアルは電子化されている。それぐらい複雑になってきたと。携帯電話ですらマニュアルはついていないですね。なので、ちゃんと電子化したもので見ないと使い方が分からないということになります。

それから、もうこれは致命的なことですが、新しいこと、個別のことには、これ、対応には限界があると。それぞれ一人一人のユーザーに対応できるようなものをすべて詰め込んでいると、ますます複雑化するわけです。ですから、ある程度需要のあるもの、別の言い方をすると、売れ行きにプラスになるものは入れる、あまり関係のないものはもう取り入れないということになります。

私がプログラミングを重視する方向に変わったのは、このトラブルですね、バグ、メーカー任せと書いていますが、バグが起こったときには、メーカーに連絡してもすぐ対応してくれるとは限らないわけです。それは非常に困る。私が困ったのは、実はこのメーカー任せではなくて、私の時代、40年ほど昔ですけれども、京大の文学部にいましたから、文学部で大型計算機を使う院生というのはあまりいなかったのですね。珍しかった。社会学と地理学と心理学ぐらいしか使いませんから。そうすると、まだ監督する教授は、よく分かっていないのですね、計算機を使うとはどういうことか。どういうことかと言いますと、使い放題だったわけです。予算の制約がなかった。いくら使っても怒られなかった。それで好きに使っていたわけですけれども。好きに使っていても、もっとどんどん使ってくださいとか、それからアルバイトで回ってきたときは、バグのチェックで、大型計算機センターから回ってきたときには1日10万円以上使いましたけれども、それでもまだ、もっと使ってバグを見つけてくれと、そのような状態でした。

大型計算機は、研究用なのでけっこう最新の、できたてほやほやのものが用意されていたのですが、数値計算のマニュアルは、情報系用にいろいろ出ていますが、あれは工学部が使うような関数のときにはあまり問題ないようですが、心理学で使う場合は、たいていどこかで止まってしまうのですね。それで、止まってしまったらエラーメッセージを見たりしますが、結局最後には対応ができないというように、もうあきらめてしまうことになるわけです。それならもう最初から自分でつくると、トラブルが起こったときにどこがどうなっているのか、まだ調べやすいわけです。そういうことがありましたので、就職先が金沢大学になったときに、この大型計算機センターとか、あるいは金沢大学の計算機センターが用意しているようなライブラリーはもう使わないと決めて、全部、一から組む方針に変えました。だからこのトラブルがメーカー任せというのが、私がプログラミングを重視した一番の理由です。プログラミングだと、自分のアイデアに対応して自由にプログラムが組める。それから、何と言ってもコストが安い。これは重要です。

プログラミングは大変だと言いますが、もう、だいたい1、2年組んでいると、必要なものはだいたいたまってきます。そうすると、そのたまったやつの再利用ということで、あとはそんなに大変ではないのですね。

私が使っている環境はどういうものかということですが、まず今のプログラミングは、GUI、グラフィックユーザーインターフェースというのが標準になっています。パソコンの場合ですけれども。特にウィンドウズパソコンなんかのときにはボタンをクリックしたりとか、そういうようなグラフィカルなインターフェースが提供されていて、それとのやり取りでプログラムを動かす。それとのやり取りというのは、例えばボタンを押すという、そういうもの、イベントと言われていますが、そのイベントが起こったときに、そのイベントから処理をするものが呼び出されるという、そのイベント駆動というプログラミングスタイルが中心になっていくわけです。

私はDelphiというのをずっと使っていました。つい最近このC++に切り替えました。理由は非常につまらない理由で、私の周りというか、うちの子どもがC++を使いだしたので、子どもとのコミュニケーションをやりやすくするために変えたというだけなのですけれども。おかげでDelphiのとき

に話が通じなかったのが、最近はよく通じるようになりました。

この使い方の説明は、ここをクリックしてもらうと、画像がコピー取れなかったので表示されていませんが、このような『大学生のための心理学VC++プログラミング入門』というのを、大学生を念頭に置いて用意していますが、ちょっとどのようなものか、実演してみたいと思います。

これがVC2008です。今ただでダウンロードできるのは2010ですが、2010は、これから説明することについてちょっと使いにくい面がありますので、このGUIの説明はこの2008でおこないます。

このExpress Editionというのは、趣味で使う場合には自由にダウンロードできるのですが、学生さん向けに用意されたページがあって、そこからだと、2005、2008、2010と、そういうものが、学生としての手続きが必要なのですが、それをするとこの2008も無料でダウンロードできます。その手続きがないと、普通の人は2010しかダウンロードできない状態に今なっています。GUIでプログラミングする場合には、今のところまだ2008ですると使いやすいということになります。

どのようなやり方になるかというと、まず「ファイル」で、「新規作成」ですね。そして「プロジェクト」。これをしますと、ここに、「場所」というのがありますが、これは適当に、「参照」のところを押しますと選べるようになっています。その選んだところで、一応ソリューションのなかのプロジェクトというかたちでプログラミングをおこなうという形をとっていますので、このプロジェクト名のところは適当に入れます。Sampleですね。例えばSampleと入れる。プロジェクト名を指定するとソリューション名が自動的に同じものになります。これで「OK」をすると、── これは 1 ギガヘルツぐらいのCPUですので、ちょっと遅いのですね。これでも使える。推奨は 2 ギガ以上になっていますが、1 ギガでも使えます──すると、こういうフォームが出てきます。このフォームは自動的に新規作成で用意されるもので、中にコードがいろいろあります。このようなかたちで最小限必要なものが自動的に用意されると。

ここへ、例えばボタンをつけたいというとき。ボタンをつけたいというようなときはどうするかというと、先ほどの、[デザイン]というところでこのフォームを出して、ここにボタンをつけるというわけですが、これは、このツールとして用意されているものから、例えばボタンをクリックして選んで、ここにぽんとクリックしてやるとボタンが張りつけられると。ボタンが張りつけられると、このようにそのボタンを表示するのに必要なプログラムが自動的に用意されています。ボタンをぽんと張りつけるだけでこういうものが自動的に入るというわけです。これだとボタンを張りつけただけです。

ここ、今何もないのですけれども、ボタンをクリックして何かさせたい。つまりボタンをクリックというイベントに対して実行されるものを用意したい。そういうとき、それはどうするかというわけですが、幾つか方法がありますけれども、一つは、このボタンをダブルクリックします。そうすると、こういうのがもう自動的に用意されます。これがボタンをクリックしたときに呼び出されるものだというのは、実はプログラムのほうでいきますと、ボタンのクリックという属性にこの関数を登録しているのです。それで、クリックしたときにこういう関数を実行してくださいよというのは、こういうコードというか、プログラムの部分が、ダブルクリックするだけで自動的に用意されます。

どういうことをするかというと、例えばボタンをクリックしたときに先ほど表示されたフォームに色をつけたいと。フォームはこのthisで表わされているわけですが、ここでこのように矢印をつけると、このthisのなかのどういうものを使うかという、こういうものが、コード支援機能と言いますが、出てきます。先頭の何文字か打ち込んでやると、それに対応するところのリストが映って、今BackColorというものが出ていますが、この状態でEnterを入れると、もうそれがここへ入ります。このBackColorに色の値を設定してやるとフォームの色が変わります。

例えば、色。Colorとしてあって、これはC++の約束なのですけれども、ちょんちょん（::）と入れてやると、この色として使える一覧表が出てくるのですね。ここで例えば途中まで打ってやると、それに一致する頭文字の、ざっと出てきますが、例えばBlueのところで止まっているときにEnterしてやると、Blueが入ると。すなわち、このものはクリックによって呼び出される関数であって、それが実行されると、フォームの色、BackColorというのはこのフォームの色ですが、それがブルーに変わると。そういうものがこれだけの

手順で、Visual Studioという開発環境が自動的に用意してくれる。これをコンピュータが実行できるかたちに変換するのに、ビルドということをおこなうと作業がずっと出てきますが、正常終了になれば、普通はデバッグというのを選びます。デバッグ開始。下にデバッグなしで開始というのがありますが、プログラムが万が一暴走したときに、こちらだとOSのほうの操作で止めるしか仕様がないのですね。でも、デバッグ開始で実行している場合だと、この開発環境のなかで止められる場合が多いです。多いというのは、暴走してしまってどうしようもないこともちろんあるのですけれども、そういうときはタスクマネジャーか何かを呼び出して強制的に止めるということになります。

これで実行すると、実行時のフォームがこのように表示されます。先ほどやったようにちゃんと色が変わるかというわけですが、クリックするとこのように変わってくれるというわけです。実際のプログラムの場合には、ここにいろいろやりたいソースをたくさん書くわけですが、基本的にはイベント駆動のGUIプログラミングというのはこのようなやり方でかなり自動化されている。自動化されてはいますが、プログラミングの対象となるそのものは、プログラマーがどんどん書いて指定していくことができるわけです。

それで、そのプログラミングについてこれから説明していきます。いろいろ用意しましたけれども、錯視をちょっとやってみたいと思います。ほかのものは一番最後のところでホームページを紹介してありますが、そこに実際のサンプルプログラムを用意していますので。

このプログラム自体は、もともとは2008でつくってあるのですが、現行版の2010でも動くという意味で、デモンストレーションは2010のVC++で実行します。

ミュラー・リヤーの錯視です。これを取り上げる理由は、これはボタンをこれだけ用意してあるのですけれども、このボタンに対して動く内容ですね、「コード」というのを選ぶとコードが表示されるのですけれども。これですね。これがキーを押したときに呼び出される関数として自動的に用意されるものです。それを用意したときに、その呼び込んだキーがMのキーであるかNのキーであるかによってこういう違い、二つの処理をおこなう、そういう内容のものです。

どういうことかといいますと、これは「GOボタンを押してください」とありますので、こういうふうになると。これは中央線の長さですね、これ、同じ長さに見えない人が多いと思います。これを同じ長さに見えるように調整するのに、例えばMのキーを押すと伸びると。先ほどの関数を呼び出しているわけです。Nのキーを押すと縮んでいくわけです。これくらいで同じに見えたとしたときに、このボタン、「Show」というボタンを押すと、ここの長さはピクセル、この交点の数で表したものがこれです。物理的には左のほうが長い。それでこうなっているのですね。最初に出したもの、このときが物理的に同じ長さです。物理的に同じ長さのときには同じに見えない。同じに見えるように調整すると実際には違いますよと。ミュラー・リヤーの錯視と呼ばれているわけですが、このようなプログラミングができるということです。

そのようにイベント駆動で実験制御のプログラムを組むわけですが、データの可視化ということで二つ取り上げています。そのうちの双対尺度法というのを説明したいと思います。

例えば分割表。この西里先生という方がこのように、喫煙が常時の人、時々の人、皆無の人と、コーヒー、紅茶どちらが好きですかというときに、コーヒーである、紅茶である、時によってコーヒーであったり紅茶であったりする、そういう二つのカテゴリーで回答者を分けたときにこうなります。

これに対して双対尺度法というのを適用しますと、これを実行しますと、すでにデータがあるので「Old Data」というのを、これをクリックしますが、データ、これはコーヒーのデータですので、これをクリックするとこのように読み込まれると。こういうプログラムが簡単に組めます。これで「OK」というのをクリックしてやると計算が可能になる。どういう計算をおこなうかというと、特異値分解というのをしています。出力用のファイルを指定してやって、すでにあるからこのように聞いてきますが、これをクリックしてから次のダイアログボックスが出るまでの時間が計算時間になります。すぐ出てきました。このあっという間の時間で特異値分解が済んでいます。もっと大きな行列でもこれくらいの時間です。

結果を図示する、ビジュアル化することでこういう結果が出る。このように図示できるわけですね。紅茶を飲む人というのはたばこをのまないと。

コーヒーを飲むという人とたばこを常時のむという人は関係が強いと。時にという人、ときどきというのはこういう関係にある。ともかく紅茶派でたばこは吸いませんというのは、ここのグループに比べて一つの特異な位置を占めているということがこのように見て分かるかたちで表示されるということになります。これがデータのビジュアル化ということです。

これは色についてのデータを図示したものです。これ、比喩ですね。コンピュータのCPUを脳の中のいろいろなことになぞらえて、このようなものも出されていますというわけです。

シミュレーションですね、統計の検定力とかをするときに、これは、なかなか数学的には解析が、ちょっと妙な積分とかしなければいけないので普通の人にはできないわけですが、シミュレーションという方法で評価することなら誰でも簡単にできる。そしてニューラルモデルですね、これはちょっとＰＤＰと呼ばれているモデルですけれども、実際にデモンストレーションをご覧になることはあまり機会がないと思いますので、ちょっとやってみますね。こういうプログラムも簡単に組めますというデモンストレーションになりますので。

どういうことをしているかというと、ニューロンのモデル。このモデルは1980年代に盛んにおこなわれた分野です。ニューラルネットワーク自体は現在も盛んに研究されていますが、脳の細胞は、この入力を受けて、それをまとめて出力すると。入出力特性をこのような関数で表しているわけです。それでネットワークを構成すると。入力層と出力層。入力層というのは網膜をイメージしています。その網膜に光が当たったときに興奮するしないのかたちで入力層に入力があって、それを、中間層一層を設けてそこで処理して、この三つの出力のうちどれが一番強い信号を出すかと。これが一番強い出力を出せばこれの表す内容として判断されていると。2 番目が一番強い反応をおこなっていれば、これの表すものとして判断されて、そのようなイメージです。

これが縦につながっているときに、エラーに相当するものが、この係数が上から下に式がつながっているかたちになっているので、エラーバックプロパゲーションと呼ばれています。誤差逆伝搬と、そのような名前で呼ばれているモデルです。

これは、実際にこの程度できますということです。このようなグラフィカルに出すわけですね。これで例えば第一のパターンとして、これは、上でずうっとなぞるだけで色が変わるようなプログラムにしてあります。具体的なプログラムの内容に興味のある人はダウンロードして見てください。これを第一パターンとして覚えさせる。次、例えば横ですね。横を、これを第二パターンとして覚えさせる。次は例えばこの囲ったやつですね。べつにこれ、規則的な形でなくても全然かまわないですよ。３種類のパターンを設定して、これと、これと、これとですね、設定して、これを学習させるということです。これは学習中の経過を出力しているもので、第一のパターンに対してはここが 1 に近づく、第二についてはここ、それから第三についてはここが 1 に近づいて、ほかはどんどんゼロに近づくところまで学習を進めさせたあとでこの弁別ということをおこなわせると。

例えば、まったく同じものでなくて、ちょっとずらしてやる。それで、何ですかと聞いてやると、1 ですよとくるわけですね。そして、例えば次。これは何ですかというと、2 と答えてくれると。あるいは、これでいくと 3 と答えてくれるというように覚えてくれるわけですね。まったく同じものを提示しなくてもちゃんと答えてくれる。では、どのあたりならどう間違えるかということになるわけですけれども、それは興味のある人はあとで試してみてください。

このようにプログラミングでいろいろできるということですが、これは統計の、先ほどのシミュレーションでそういうデータを取るとどの程度結果が出るかということのアルゴリズムで用意したものです。

最後に私の研究についてちょっと説明させていただくと、今まではpsychophysical laws、これはGescheiderという人が出しているパラダイムですが、単純な刺激に対して感覚がどのようなものになっているか、それをpsychophysical lawsと呼ぼうと。複数の感覚情報をどう統合しているか。それはpsychological lawsと呼ぼうと。psychological lawsというのは、この複数の感覚を統合して得られる感覚ですが、その感覚がそのまま反応に出てくるわけではないわけです。人は相手のことを考えながら、自分の考えをそのまますっと出すということは普通しない。いろいろ調整してから自分の考えなり意見を出すわけですが、それがsensory response laws、感覚と反応の関係という定義になります。

確かに、今までここの部分で主にやってきたのですが、これからちょっとここを重点的にやってみようかと考えています。それは、この複数の情報をこのようにウエイトをかけた和で統合を表すという、そういうモデルがあるのですが、大山先生らが、これとならった結果をやると、こういう単純なものではなくて、ここはちょっと文系の先生ですから、表現の仕方が弱いといえば弱いのですが、刺激の優位さですね、それを反映したモデルのほうが当てはまるということを出されています。私はここに量子論的なメカニズムを想定するともっと面白いのではないかということで、その観点からこれからちょっと調べてみようかと。昔、物理学に興味を持っていたというところがこういうところで出てくるわけですね。

その量子論的メカニズムというのは、シュレーディンガーの猫というように、量子論のときには、二つのというか、複数の状態があって、それが確率論的にどれを取るかというのが表されているわけですね。複数の状態が共存するというのは、心理学ではこのような曖昧図形とか、図と地の反転というのであるわけですが、これはどういうことかというと、これは人によって見え方が違っていると思います。ここの部分がこっち方向に飛び出てみえている人は、これが上から見た目になっていると思います。ところが、これが右上に飛び出て見えている人には、この下から見た状態になっていると思うのですね。どちらで見えているか人によって違うと思うのですが。違う見方が見えるかということで、ちょっと頑張ってもらえば違うほうの見え方ができると思います。

ね、両方見えるでしょう。え？片方しか見えない？まあ、あとで頑張ってください。

これはルビンという、これ、英語だとgobletで、盃と書いているのが多いのですが、日本人の解説だとvaseで花瓶になっているのですね。どこでどう食い違ったのかそこのところはあれですが。

これは人の顔に見える場合と、これが盃に見える場合とあります。これ、人の顔に見えている人はどれくらいいますか。うん。人ではなくて盃に見えている人は。ね、両方いるでしょう。でも、同時に二つは見えない。いや、見えているという人はいますか。いないでしょう。だからこれ、見えているほうを図というのですけれども、顔が図になる場合と、盃が図になる場合があって、これ、見ているとどんどん入れ替わっていくわけです。これを量子論的な考え方で記述してやっていきたいということを今考えております。

ざっと駆け足で説明しましたが、サンプルプログラム自体は、Delphiについてはこちらですね。これは大学のホームページに授業関係のページとかいろいろ上げているのですが、ここをクリックするとサンプルプログラムがアップロードされているページに移ります。ここに主にDelphiのものを上げてあります。授業の関係でほかの言語も上がっていますけれども。それから、これは最近、子どもとの会話をやりやすくするためにCを始めて、これも利用することになったわけですけれども。これはニフティのほうのですね。これはもうC++関係、特にVC++のサンプルプログラムを中心に上げてあります。ここをクリックしていただくと、そのページに入ります。そういうことですので、関心のある人はちょっと試してみてください。もし何かあればＥメールをいただけば、できるだけ答えられることは答えるようにします。

では、簡単ですけれども、これで。