令和元年 11 月 19 日

京都工芸繊維大学コンピュータ部

Lime 60

はじめに

コンピュータ部部長の寺村です。この度はお忙しい中、本学の学園祭にお越しいただき、我々コンピュータ部にご興味、ご関心を持っていただきありがとうございます。そして、この Lime60 号を手にしてくださったことを、部を代表いたしまして厚く御礼申し上げます。この Lime は部員の活動の一部を記したものです。日々の活動の成果や部員個人の趣味など、内容は多岐にわたります。これを通じて、コンピュータ部のことを知っていただき、少しでも我々の活動に興味を持っていただければ幸いです。そして、今回の Lime 作成にあたり編集を担当した山上君、記事を執筆した部員各位、ならびに OB・OG の方々を含め、日々我々の活動を支えてくださっているすべての方々への感謝をもって始めの挨拶とさせていただきます。

令和元年 11 月 14 日

京都工芸繊維大学コンピュータ部副部長　寺村 英之

目次

はじめに iii
1　シリコンバレー旅行記　—　山上 健 1
2　Sigfox ネットワークで環境情報を集めてツイートする話　—　寺村 英之　6
3　SVM 勉強ノート　—　兼光 琢真 13
4　ラスタースクロールを再現しよう　—　上北 裕也 25
編集後記 32

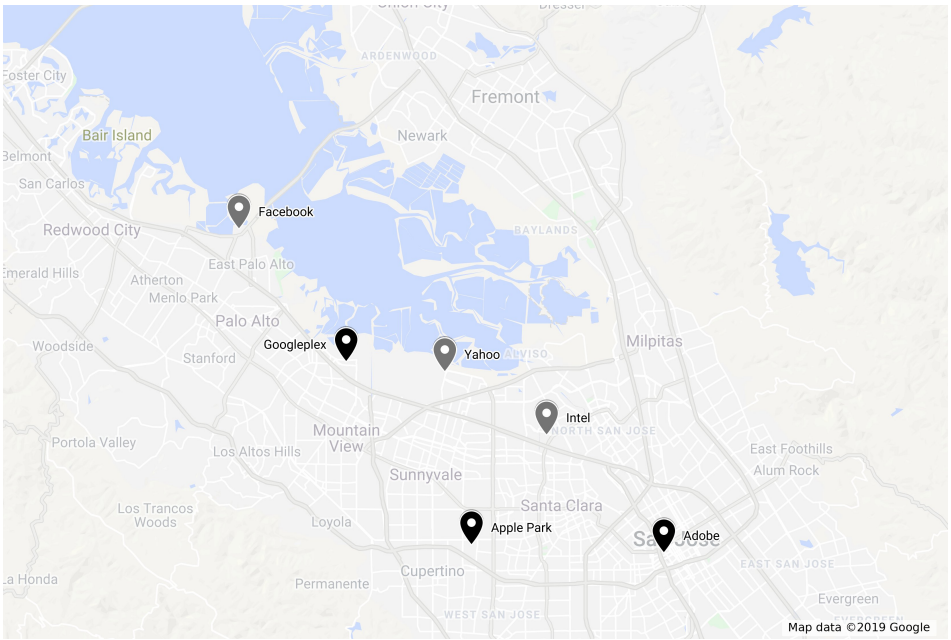
1 シリコンバレー旅行記

機械課程 2 回生 山上 健

1.1 はじめに

こんにちは。前回の Lime で Python のワンライナーを紹介した、山上です。10 月初めに家族とシリコンバレーに行ってきたので、今回はその話をしようと思います。旅行そのものは数日でしたが、単独で行動した 2 日目の内容のみお話しします。

先にシリコンバレーについて説明すると、アメリカ合衆国カリフォルニア州の、コンピュータ関連企業が多く集まっている地域のことです。



1.2 ホテル ～ Adobe 本社ビル

当日の朝まではホテルのネットワークを使って Youtube でも見ながら一日を過ごそうと思っていましたが、「せっかくシリコンバレーに来たし有名企業の社屋をめぐってみよう」と重い腰を上げたのがすべての始まりでした。

まず、Adobe の本社ビルへ向かいました。移動には **Uber** という配車サービスを利用しました。Uber を一言で表すと、一般人が自家用車で送迎してくれるタクシーのようなものです。これがなかなか便利で、アプリで乗車位置と降車位置を指定すると、近くを走っている車が数分で迎えに来てくれます。運賃とチップはクレジットカードで支払うので、現金のトラブルの心配がありません。(僕はこの旅で親のクレジットカードを決済に使ったのですが、サービスの快適さにチップをはずみすぎて後で親に怒られました。)

降車後、空気に馴染むため周辺を探索。比較的背の高いビルが多い町並みでしたが、道が広いので、風がよく通るうえに日光が届いてとても過ごしやすい雰囲気でした。

Adobe の本社ビルの前で写真を撮り、時間を確認すると 12 時を過ぎていました。ご飯を買おうとセブンイレブンに入るや否や、店員 1 人に 10 人くらい並ぶ光景を目にして即

刻退店。どこか飲食店で食べようかとも思いましたが、日本でも知らない店は避けたい性格の僕は、英語で上手く注文できるか不安もあって、最終的に昼食を抜くことを決めました。

1.3　Adobe 本社ビル ～ Apple Park

次に Apple Park(Apple 本社敷地) へ向かいました。手持ちのガイドブックを参考にして、バスに乗ることを決意。異国の乗り物を独りで、しかも初めて利用するというのはかなり勇気が要ることでしたが、案ずるより産むが易し、特に問題なく乗り降りできました。ちなみにそのバスは、前の扉から乗り、降りる際は壁をつたうロープを引っ張る仕様でした。

Apple 本社ビルのある区域は一般客は立ち入ることができませんが、その代わりに道を挟んで向かい側にビジターセンターという来訪者向けの建物があります。内観は日本の Apple ストアとほぼ同じでした。

このビジターセンターの名物は、建物の一角にある、Apple Park 全体の模型です。iPad のカメラをそれに向けると AR(拡張現実) が楽しめると聞いていたので、持参した iPad をかざしてみました。しかし何も起きません。スタッフが持つ特別な iPad が必要だったのです。自分から話しかけようとスタッフに近づいてみると「Do you wanna try？」と先手を打たれ、この日初の英語が「Yes」に。

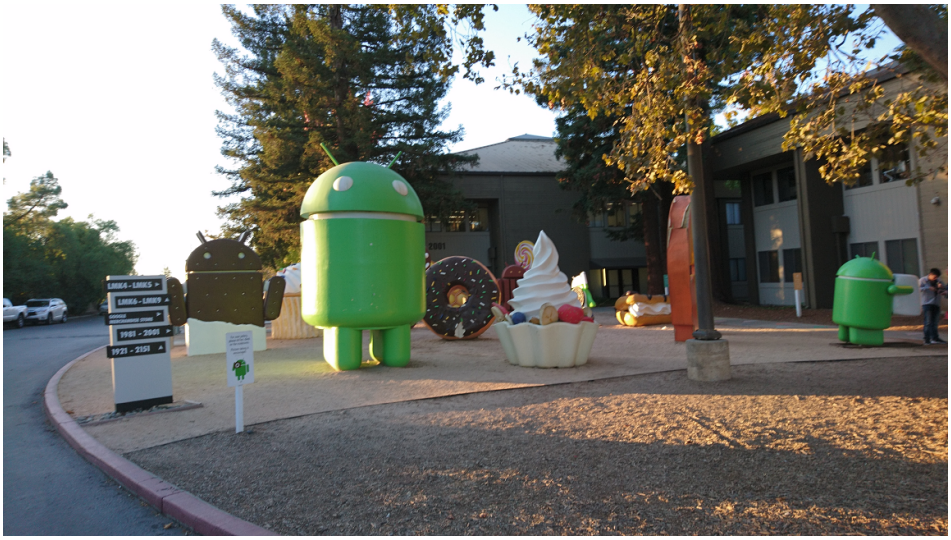
ここでご当地グッズを買うかどうか悩むこと 1 時間。というのも、品物が T シャツとトートバッグと帽子くらいしかなく、知人にお土産として配れるようなものがなかったのです。結局購入は諦め、次に向かう Googleplex(Google 本社敷地) に期待して Apple を後にしました。

1.4 Apple Park ～ Googleplex

この時点で 15 時半くらいだったので急いで Googleplex へ。Google マップで経路を検索し、再びバスに乗ることに。バス経験者の僕には造作ないことでした。

Googleplex は一般人でも立ち入れることができ、どこか大学のような雰囲気でした。どの建物も入れそうな感じがしましたが、実際に入れたのはグッズショップのみ。ショップの前には Android のマスコットキャラクターがいっぱいいました。

ここの品揃えは豊富でした。服や鞄に加えて、文房具、シールやおもちゃなどがあり、逆に選ぶのに時間がかかりました。



1.5　Googleplex ～ ホテル

お土産を何点か買い、店を出ると日が沈みかけていました。他に Yahoo や Intel なども見に行きたかったのですが断念して、Uber でホテルへ。

半日歩き回ったうえに昼食を取っていなかったのでさすがに空腹でした。ホテル付近の飲食店はマクドナルドとサブウェイのみ。どちらも日本で展開しているチェーン店であり味は知っているので、個人的に好きな方のサブウェイを選択しました。しかし、それが間違いだったのです。サブウェイといえばサンドイッチのパンの種類やトッピングを自分で選べることで有名です。そのシステムをてっきり忘れていた僕は、どのトッピングがいいか逐一聞いてくる店員の英語に対処しきれず、ただ野菜の名前を列挙することしかできませんでした。

日本でも注文が難しいサブウェイをなぜ選んでしまったのか。それまでアプリや交通機関を使い上手く立ち回れていただけに、最後の判断ミスに少し後悔がありましたが、サンドイッチは予想通り満足できる味でした。一日の達成感に浸りながら僕は眠りにつきましたとさ、めでたしめでたし。

1.6　おわりに

このシリコンバレーの旅を通じて、自分で判断すること、自発的に行動することの意味を学ぶことができました。結果、刺激的で充実した旅になりました。最後まで読んでいただきありがとうございました。

2 Sigfox ネットワークで環境情報を集めてツイートする話

情報課程 3 回生 寺村 英之

2.1 はじめに

どうも、KITCC 部長の寺村です。この記事では 4 月のはじめ辺りから触っていた Sigfox ネットワークを使ってそれらしいことができるようになったので紹介しようと思います。

2.2 最近 Twitter Developer アカウントを取りました

最近 Twitter Developer[*1]にサインアップしました。Twitter Developer アカウントを取得すると Twitter API にアクセスできるようになり、過去の様々なツイートを集めたり、自分で作ったプログラムからツイートができるようになります。例えばこのようなプログラムでツイートができます。

```
1  import os
2  import requests
3  from requests_oauthlib import OAuth1Session
4
5  consumer_key = os.environ['TWITTER_CONSUMER_KEY']
6  consumer_secret = os.environ['TWITTER_CONSUMER_SECRET']
7  access_token = os.environ['TWITTER_ACCESS_TOKEN']
8  access_token_secret = os.environ['TWITTER_ACCESS_TOKEN_SECRET']
9
10 param = requests.urllib3.request.urlencode({ "status" :
```

[*1] https://developer.twitter.com/

```
    status_str })
11 url = 'https://api.twitter.com/1.1/statuses/update.json?{}'.
    format(param)
12
13 session = OAuth1Session(consumer_key, client_secret=
    consumer_secret, resource_owner_key=access_token,
    resource_owner_secret=access_token_secret)
14 session.post(url)
15 session.close()
```

Twitter API を使って投稿（status の更新とも言われる）するには、OAuth を使った認証を行う必要があります。ツイートをする Twitter ユーザとして投稿をするのでそのユーザのアクセストークンと Twitter App のコンシューマキーが必要です。OAuth における一連のやり取りは様々な言語向けにライブラリが開発されているのでそれを用いれば比較的シンプルに OAuth を使った API を呼び出すことができます。

自分で作ったプログラムからツイートできるとなると、Twitter ボットを作りたくなってきます。私はよくマイコンボードを使って開発することがあるので、マイコンボードから取得した周辺温度や雷の活動などを記録してツイートするシステムを作ってみることにしました。

2.3 作ってみる

作ったシステムは温度などを測定してインターネット上に送信する「観測システム」と受信したデータを蓄積して定期的にツイートする「ツイートシステム」の 2 つからなりま

す。ツイートシステムは自分で用意したサーバ上で動作するプログラムからなり、この部分は前の項で紹介したプログラムに外部からの通信を受けて情報を記録する機能を付けた程度のもので、今の所面白みはあまりないためこの記事では触れるだけにします。観測システムはマイコンボードを中心に接続されたセンサで得られたデータを定期的にインターネット上のツイートシステムに送信するものです。最終的に太陽電池で駆動することを目標にするためにできるだけ少ない電力で動くように作ることにしました。

消費電力を抑えるためには回路の工夫も当然必要ですが、インターネット通信に用いるインタフェースも考える必要があります。以下でその周辺の話題について述べます。

Wi-Fi と（一般的な）モバイルデータ通信は消費電力が大きめ

近くに無線 LAN アクセスポイントがあるなら Wi-Fi 通信モジュールを用いてデータを送受信できます。また 3G 通信モジュールなどを用いれば携帯電話回線を使って通信することもできます。これらを用いた通信はパソコン上のプログラムでインターネットを介した通信を行うこととほとんど同じで、適切なライブラリを用いてアプリケーション層の通信要求を発行するだけで通信できます。一方でこれらの通信様式は通信に必要なリソースが多くなる代わりに通信帯域やリアルタイム性を重視しているので、通信モジュールの消費電力量が大きくなりがちです。例えば無線 LAN 通信モジュールを搭載している ESP32-DevKitC では通信時に実測で 5V 200mA 程度継続的に消費します。また何もしていない間でもいくらか電力を消費します。太陽電池を用いている場合など電力量が限られている場合は通信していない間に通信部の電源を切るなどの工夫が必要になります。

他にも Bluetooth 通信モジュールを使って間接的に通信する方法（こちらはインターネットへデータを送るゲートウェイを作ることになる）などがありますが、今回はそれらとは別の Sigfox ネットワークを使って通信してみることにしました。

Sigfox ネットワーク

Sigfox は帯域が小さい代わりに低電力で動作する通信モジュールを作ることができる広域無線通信規格です。Sigfox ネットワークはそれを用いた通信ネットワークで通信機器（例えば通信モジュール）ごとにどのインターネット上のサーバと通信するか決めることができます。これを用いることでインターネットへのセンサなどの観測データの送信を簡単にかつ少ない消費電力で実現できます。一方で先程挙げた通信方法と異なり、帯域が小さいだけでなく通信方法にも制限があります。まず、無線 LAN などのように TCP や UDP を用いた通信を行うことはできません。Sigfox では小さい帯域をできるだけ活用するために通信機器ごとにどのようにデータを解釈するかを自分で決めることになります。つまり直接 Twitter API を呼び出すことはできません。さらに、通信機器へ向かってインターネットから能動的にデータを送ることはできません。下り通信を行うにはまず通信

機器側から下り通信要求をデータとともに送る必要があります。

このように Sigfox ネットワークでの通信はかなり厳しい制約がありますが、周辺温度の定期的な送信＋時刻の受信やセンサの値が規定値を超えた場合のデータ送信など通信機器主導かつデータ量が比較的少ない通信目的の場合はあまり気になりません。今回作りたいシステムはまさに当てはまっているので少ない犠牲で消費電力を下げることができます。

2.4　mbed で Sigfox 通信モジュールを使ってみる

観測システムの試作にはマイコンボードとして Seeed の Arch Pro[*2]を、Sigfox 通信モジュールには BRKWS01[*3]を使いました。Sigfox 通信モジュールは UART で AT コマンドをマイコンボードから送ることで制御します。AT コマンドはテキストデータからなるのでシリアルコンソールにメッセージを送るのと同じような感覚で制御プログラムを実装できます。すでに mbed 対応マイコンボード向けに Sigfox モジュール向けの AT コマンドライブラリがいくつかアップロードされていましたが、せっかくなので自分で実装することにしました。

2.4.1　データの送信

データの送信には `AT$SF` コマンドを使います。このコマンド名のあとに `=` と十六進数の表記に変換したペイロード（送信データ）をつないで最後に CR（モジュールによるかもしれない？）を送ります。例えば 2 バイトのデータ `0x01, 0x7F` を送りたいときは次のようにします。

```
1  char data[5] = "017F";
2  serial->printf("AT$SF=%s\r", data);
```

送信に成功するとモジュールから `OK` と送信されてきます。

[*2] http://wiki.seeedstudio.com/Arch_Pro/
[*3] https://www.kccs-iot.jp/solution/product/device32/

2.4.2　データの送信と受信

先程のコマンドに ,1 をつけることでデータの送信時に下り通信を要求することができます。受信したデータはモジュールから UART で送られてきます。

```
1  serial->printf("AT$SF=%s,1\r", data);
2  while(!this->serial->readable());
3  // これ以降で受信データをパースする
```

これらのコマンドを用いて BME280 環境センサで観測した周辺温度、湿度、気圧を 15 分毎に送るようにプログラムを作成しました。それぞれのデータは生の値だと 32 ビット浮動小数点数で得られますが、これでは 1 度の送信で間に合わず 1 日の通信制限に引っかかってしまうので次の式で精度を落として 12 バイトから 3 バイトにデータ量を落としました。

データ種別	変換式
温度	x + 40.0 を uint8_t へ（小数点以下切り捨て）
湿度	x を uint8_t へ
気圧	1100.0 - x を uint8_t へ

2.5　動作させてみる

試作した観測システムを次の写真に示します。ツイートシステムを動かすと送信されてくるデータが蓄積されて定期的にツイートされます。

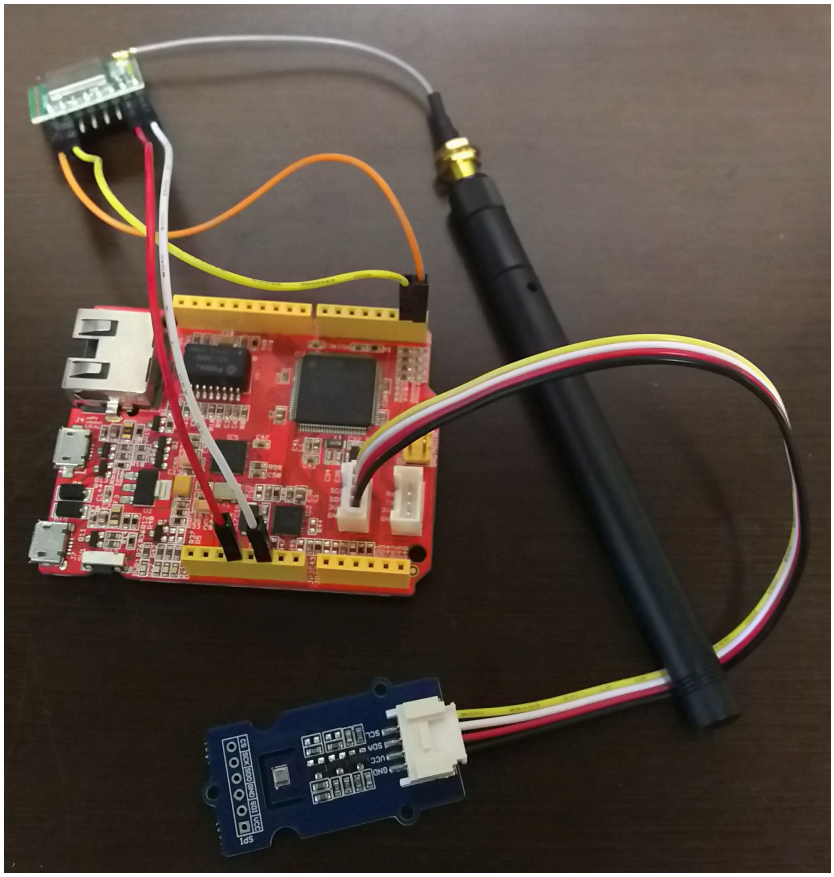


図 2.1　観測システムの試作機

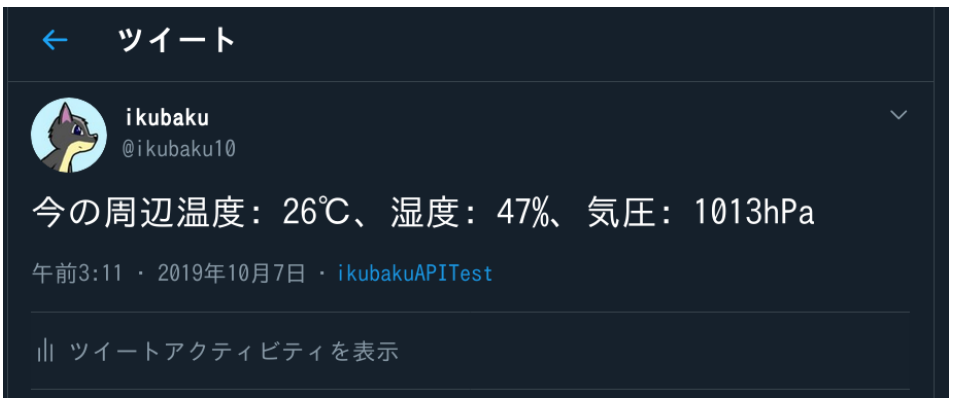


図 2.2　ツイートの様子

2.6　今後やりたいことと課題

2.6.1　専用ハードウェアの作成

試作で用いたマイコンボードは少々オーバースペックで、もっと低消費電力のものを使う余地があります。またセンサの種類を増やしたり太陽電池などから電力を得たりするよ

うにするには自分で設計した回路がある方が便利ですし、外で動かすにはケースが必要です。これから専用のハードウェアを練ることができたら良いと思っています。

2.6.2　通信のフェイルセーフ機構を実装する

観測システムのプログラムでは通信モジュールとの通信失敗時のフェイルセーフ機構を作っていなかったので時々動作が止まってしまいました。今後ライブラリを改善してタイムアウト処理を実装したり、自動的に通信モジュールを再起動する機構を用意したいと考えています。

2.6.3　センサの追加

今は定期的に情報を観測することしかしていないので、雷センサを新たに追加して突発的な事象について反応できるようにしたいと思っています。他にも紫外線センサや二酸化炭素センサなどを搭載して見ていて面白いボットが作れたら良いと思います。

2.7　おわりに

これまでマイコンボードで作ったシステムとインターネット上のプログラムを連携させて動かしてみたことがなかったので、システムの実装と実際の動作の確認を楽しんですることができました。まだ理解が浅いので具体的な作り方をここでは紹介できませんでしたが、今後もっと詳しい話を紹介したりより複雑なシステムを作って公開できたら良いと思っています。

現在作成したシステムは止まっていますがフェイルセーフ機構の実装と専用ハードウェアの作成ができたらボット用のアカウントを作成してもう一度動作させるつもりです。ほかにも Sigfox モジュールの制御用ライブラリなどの副産物も紹介できたら良いと思っています。

この記事で紹介したシステムの開発や他の話題を私の Twitter アカウント `@ikubaku10` やブログ `https://ikbk.net/` で紹介しているので気になる方はぜひご覧ください。

3 SVM勉強ノート

情報工学課程 3 回生 兼光 琢真

3.1 はじめに

SVM(Support Vector Machine) は 2 クラス分類問題に対する代表的な手法です．SVM の目的は，図 3.3 のような直線 (一般には超平面) を見つけることです．この直線は制約付き凸 2 次計画問題という種類の最適化問題の最適解として定まります．この種類の最適化問題は汎用的な解法が知られていますが，訓練データの数が多くなるほど計算時間が莫大に長くなります．

そこで，SMO(Sequential Minimal Optimization) という SVM のための最適化アルゴリズムを用いて最適解を求めます．SMO は SVM で解くべき最適化問題の特徴をうまく利用しています．

SVM は線形分離できる (ある超平面で分離できる) 訓練データに対しては，分離できる超平面を必ず求めることができます．一方，線形分離できない問題もありますが，その場合にはカーネル法が有効かもしれません．

カーネル法のコンセプトを説明します．元のデータに非線形変換 Φ を施せば，もっと分離しやすくなる場合があります．そこで，元の空間を Φ で曲げた後に SVM で分類するのです．

3.2 最適化問題の導出

n を訓練データの数とし，$1 \leq i \leq n$ に対して $x_i \in \mathbb{R}^p$，$y_i \in \{-1, 1\}$ とします．目標は，

$$f(x) = w^\top x + w_0$$

の形の関数を作って，$f(x) > 0$ ならば $y = 1$，$f(x) < 0$ ならば $y = -1$ と分類することにします．そのために，適切な $w \in \mathbb{R}^d$，$w_0 \in \mathbb{R}$ を求めます．

3.2.1 線形分離できる場合

まずは，超平面 $w^\top x + w_0 = 0$ と x_i との距離が

$$\frac{|w^\top x + w_0|}{||w||} \tag{3.1}$$

になることを説明します．x_i から超平面への垂線を考えると，その垂線の足はある実数 k により $x_i + kw$ で表せます．これが超平面上にあるのですから，

$$w^\top (x_i + kw) + w_0 = 0$$

が成立し，超平面との距離 $|kw|$ を式 (3.1) に変形できます．

線形分離できるときを考えているので

$$^\exists M \in R_{>0}, \quad ^\forall i, \quad y_i f(x_i) \geq M \tag{3.2}$$

が成立します．

これを踏まえて，最適な超平面を求めるために次の最適化問題 (3.3) を解きます．

$$\begin{aligned} &\max_{w,w_0,M} \frac{M}{||w||} \\ &\text{s.t.} y_i(w^\top x_i + w_0) \geq M \end{aligned} \tag{3.3}$$

最大化のため M を大きくしようにも，制約で点 x_i と超平面の距離よりは大きくできません．式 (3.2) より必ず正数 M が存在し，そのときすべての点は分離されているから，結果的に M は超平面とそれに一番近い点との距離に一致します．

更に $\tilde{w} = w/M$，$\tilde{w_0} = w_0/M$ とすれば，最適化問題 (3.3) を次の形に変形できます．

$$\begin{aligned} &\max_{\tilde{w},\tilde{w_0}} \frac{1}{||\tilde{w}||} \\ &\text{s.t.} y_i(\tilde{w}^\top x_i + \tilde{w_0}) \geq 1 \end{aligned} \tag{3.4}$$

この最適解から求められる超平面は，元の最適化問題 (3.3) で求まる超平面に一致するので，変数 M を消去したこの最適化問題を解くことにします．以下，$\tilde{w}$，$\tilde{w_0}$ の代わりに w，w_0 を使います．また，$1/||w||$ の最大化は $||w||^2$ の最小化と同じことなので，以下では $||w||^2/2$ の最小化問題を考えます．ここの 2 乗や 2 で割るのは，最適化を考えやすくするためです．

3.2.2 線形分離できない場合

線形分離できない場合でも，マージン (超平面と点との間の空間) の内側に点が入ることを許して，SVM を適用させられるようにします．この考えで最適化問題 (3.3) の制約を緩和する自然な方法として，新たにマージンのゆるさを表す変数 $\xi_i > 0$ を導入して

$$y_i(w^\top x_i + w_0) \geq M - \xi_i$$

とする方法があります．このまま最適化問題 (3.4) に変形すると扱いづらい非凸最適化問題になってしまうので，$\tilde{\xi_i} = \xi_i/M$ としておけば次の最適化問題 (3.5) になります．

$$\begin{aligned} \min_{w,w_0,\xi} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ \text{s.t.} & y_i(w^\top x_i + w_0) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{3.5}$$

ただし，C は正の実数で事前に与えます．目的関数に追加された第 2 項には，ξ_i を小さく保ちたい気持ちが込められています．C を大きく与えるほど ξ_i を増加させたときの影響が大きいので，より ξ_i を小さく保ちたいということになります．$C = \infty$ とすれば，ξ_i を 0 より少しでも大きくしてしまうと目的関数の値も ∞ となるので，常に $\xi_i = 0$ とせざるを得なくなり，最適化問題 (3.4) に一致します．なお，第 2 項が無ければ (つまり $C = 0$ ならば)，マージンのゆるさを表す ξ_i を限りなく大きくすることでどんな超平面でも制約を満たせてしまいますので，デタラメな結果になるでしょう．

3.2.3 双対定理

【元問題】

$$\begin{aligned} \min_x & f(x) : \mathbb{R}^d \to \mathbb{R} \\ \text{s.t.} & h_i(x) = 0 \\ & g_j(x) \leq 0 \end{aligned} \tag{3.6}$$

に対するラグランジュ関数を，$\lambda_i \in \mathbb{R}$，$\mu_j \in \mathbb{R}_{\geq 0}$ を導入して

$$L(x, \lambda, \mu) = f(x) + \sum_i \lambda_i h_i(x) + \sum_j \mu_j g_j(x)$$

と定めます．ここで元問題 (3.6) に関連した 2 つの最適化問題を考えます．

【主問題】

$$\inf_x \sup_{\lambda,\mu} L(x, \lambda, \mu) \tag{3.7}$$

【双対問題】

$$\sup_{\lambda,\mu}\inf_{x} L(x,\lambda,\mu) \tag{3.8}$$

主問題 (3.7) の sup する部分を見ますと

$$\sup_{\lambda,\mu} L(x,\lambda,\mu) = f(x) + \sup_{\lambda}\sum_{i}\lambda_i h_i(x) + \sup_{\mu}\sum_{j}\mu_j g_j(x)$$

で，第 2 項と第 3 項は

$$\sup_{\lambda}\sum_{i}\lambda_i h_i(x) = \begin{cases} 0 & (h_i(x)=0) \\ \infty & (\text{otherwise}) \end{cases}$$

と

$$\sup_{\mu}\sum_{j}\mu_j g_j(x) = \begin{cases} 0 & (g_j(x)\le 0) \\ \infty & (\text{otherwise}) \end{cases}$$

なので，実行可能領域を $\mathcal{F}$ とすれば

$$\sup_{\lambda,\mu} L(x,\lambda,\mu) = \begin{cases} f(x) & (x\in\mathcal{F}) \\ \infty & (\text{otherwise}) \end{cases}$$

です．つまり，元問題 (3.6) の最適解が存在すれば，主問題 (3.7) の最適値に一致します．

次に，任意の x,λ,μ に対して，λ,μ を固定すれば

$$\inf_{x} L(x,\lambda,\mu) \le L(x,\lambda,\mu)$$

が成立し，一方で x を固定すれば

$$L(x,\lambda,\mu) \le \sup_{\lambda,\mu} L(x,\lambda,\mu)$$

が成立します．まとめると

$$\inf_{x} L \le L \le \sup_{\lambda,\mu} L \begin{cases} = f & (x\in\mathcal{F}) \\ = \infty & (\text{otherwise}) \end{cases} \tag{3.9}$$

が成立して，これは弱双対定理と呼ばれます．

ある点 (x^*,λ^*,μ^*) を決めたとき，この点が L の鞍点であるとは

$$L(x^*,\lambda,\mu) \le L(x^*,\lambda^*,\mu^*) \le L(x,\lambda^*,\mu^*)$$

が成立することを指します．このとき次が成立します．

$$\sup_{\lambda,\mu}\inf_{x} L = L(x^*,\lambda^*,\mu^*) = \inf_{x}\sup_{\lambda,\mu} L \iff (x^*,\lambda^*,\mu^*) \text{ が } L \text{ の鞍点} \tag{3.10}$$

[⇒] の証明:

$$L(x^*,\lambda,\mu) \overset{\text{仮定}}{=} \inf_x L \leq \sup_{\lambda,\mu}\inf_x L \overset{\text{仮定}}{=} L(x^*,\lambda^*,\mu^*) \overset{\text{仮定}}{=} \inf_x \sup_{\lambda,\mu} L \leq \sup_{\lambda,\mu} L \overset{\text{仮定}}{=} L(x,\lambda^*,\mu^*)$$

[⇐] の証明: 仮定 $L(x^*,\lambda,\mu) \leq L(x^*,\lambda^*,\mu^*)$ はつまり

$$\sup_{\lambda,\mu} L(x^*,\lambda,\mu) = L(x^*,\lambda,\mu)$$

のことであり，同様に仮定より

$$\inf_x L(x,\lambda^*,\mu^*) = L(x^*,\lambda,\mu)$$

でもありますので，

$$\inf_x \sup_{\lambda,\mu} L \leq \sup_{\lambda,\mu} L = L(x^*,\lambda^*,\mu^*) \leq \inf_x L \leq \sup_{\lambda,\mu}\inf_x L$$

が成立します．一方，弱双対定理より $\inf_x \sup_{\lambda,\mu} L \geq \sup_{\lambda,\mu}\inf_x L$. □

L の鞍点 (x^*,λ^*,μ^*) が存在して，L は鞍点で微分可能であるとすれば，x^* は $\min_x L(x,\lambda^*,\mu^*)$ の最適解だから

$$(\nabla_x L)(x^*,\lambda^*,\mu^*) = 0$$

が成立します．また，λ^*,μ^* は次の最適化問題 (3.11)

$$\begin{aligned} &\max_{\lambda,\mu} L(x^*,\lambda,\mu) \\ &\text{s.t.}\mu_j \geq 0 \end{aligned} \tag{3.11}$$

の最適解，つまり変形した次の最適化問題 (3.12)

$$\begin{aligned} &\min_{\lambda,\mu} -\sum_i \lambda_i h_i(x^*) - \sum_j \mu_j g_j(x^*) \\ &\text{s.t.} -\mu_j \leq 0 \end{aligned} \tag{3.12}$$

の最適解なので，KKT 条件 (説明は省略しますが，ここでは目的関数の値が $-\infty$ にならないように条件を定めると一致する気がします)

$$\begin{aligned} h_i(x^*) &= 0 \\ \mu_j \geq 0, g_j(x^*) &\leq 0, \mu_j g_j(x^*) = 0 \end{aligned}$$

が成立します．以上をまとめると，L の鞍点は KKT 条件を満たすことが分かりました．さらに，凸計画問題であれば適当な制約想定のもとで逆も成り立つので，

$$(x^*,\lambda^*,\mu^*) \text{ が KKT 条件を満たす} \iff (x^*,\lambda^*,\mu^*) \text{ が } L \text{ の鞍点}$$

が成立して，これは強双対定理と呼ばれます．

原稿の締め切りがとうに過ぎているので，KKT 条件の周辺の理解を残念ながら打ち切りました．理解が中途半端なので，正しくない部分があると思います．理論展開は [1] の 4 章を参考にしましたので，詳細について興味を持たれた方は参照してみてください．

3.2.4 双対問題

SVM の場合は双対問題にすると制約条件が簡単になって最適化しやすくなります．最適化問題 (3.5) は適当な凸計画問題なので，強双対定理等より双対問題の最適解が元問題の最適解に一致します．そこで，双対問題の KKT 点を探し，求めたかった超平面のパラメータ w, w_0 を計算する問題に帰着させます．

まず，最適化問題 (3.5) を次の形に書き換えます．

$$\begin{aligned} \min_{w,w_0,\xi} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ \text{s.t.} & -(y_i(w^\top x_i + w_0) - (1-\xi_i)) \leq 0 \\ & -\xi_i \leq 0 \end{aligned} \tag{3.13}$$

この問題に対するラグランジュ関数は，正の実数をとる変数 α_i，μ_i を導入して

$$L(w, w_0, \xi, \alpha, \mu) = \frac{1}{2}||w||^2 + C\sum_i \xi_i - \sum_i \alpha_i(y_i(w^\top x_i + w_0) - (1-\xi_i)) - \sum_i \mu_i \xi_i$$

になります．

書き換えた最適化問題 (3.13) の双対問題は

$$\begin{aligned} \max_{\alpha,\mu} \min_{w,w_0,\xi} & L(w, w_0, \xi, \alpha, \mu) \\ \text{s.t.} & \alpha_i \geq 0 \\ & y_i(w^\top x_i + w_0) - (1-\xi_i) \geq 0 \\ & \alpha_i(y_i(w^\top x_i + w_0) - (1-\xi_i)) = 0 \\ & \mu_i \geq 0 \\ & \xi_i \geq 0 \\ & \mu_i \xi_i = 0 \end{aligned} \tag{3.14}$$

です．目的関数 L は w, w_0, ξ に関して凸関数なので min する部分に注目すると，最適解において

$$\begin{aligned} \frac{\partial L}{\partial w} &= w - \sum_i \alpha_i y_i x_i = 0 \\ \frac{\partial L}{\partial w_0} &= -\sum_i \alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} &= C - \alpha_i - \mu_i = 0 \end{aligned} \tag{3.15}$$

が成り立つことが分かります．これを最適化問題 (3.14) に代入して整理すると

$$\begin{aligned} \min_{\alpha} & \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C \end{aligned} \tag{3.16}$$

になります．

この最適化問題 (3.16) の最適解 α が求まったとします．すると，w は式 (3.15) の第 1 式より

$$w = \sum_i \alpha_i y_i x_i$$

で求められます．

次に w_0 の求め方を説明します．もしも $0 < \alpha_i < C$ ならば式 (3.15) の第 3 式より $0 < \mu_i < C$ であり，さらに最適化問題 (3.14) の制約条件第 6 式より $\xi_i = 0$ が成立します．このとき $\alpha \neq 0$ かつ $\xi_i = 0$ なので，制約条件第 3 式より

$$y_i(w^\top x_i + w_0) - 1 = 0$$

でが成立します．つまり，$0 < \alpha_i < C$ を満たす適当な i を用いて w_0 を

$$w_0 = \frac{1}{y_i} - w^\top x_i = y_i - \sum_j \alpha_j y_j x_j^\top x_i$$

と求めることができます．実装では，別々の i から計算した複数の w_0 を平均します．

3.3 SMO アルゴリズム

解くべき最適化問題 (3.16) に対して $\beta_i = \alpha_i y_i$，$K_{ij} = x_i^\top x_j$ と定義して書き直せば

$$\begin{aligned} \min_{\beta} & \frac{1}{2}\sum_{i,j=1}^{n}\beta_i\beta_j K_{ij} + \sum_{i=1}^{n}\beta_i y_i \\ \text{s.t.} & \sum_{i=1}^{n}\beta_i = 0 \\ & 0 \le \beta_i \le C \qquad (y_i = 1) \\ & -C \le \beta_i \le 0 \qquad (y_i = -1) \end{aligned} \tag{3.17}$$

と表せます．

SMO アルゴリズムでは，この最適化問題 (3.17) に対して 2 変数 β_s, β_t 以外を固定して最適化することを繰り返します．

更新量を d として $\beta_s \leftarrow \beta_s + d$ と更新したとき，最適化問題 (3.17) の制約条件第 1 式より $\beta_t \leftarrow \beta_t - d$ と更新しなければなりません．さらに，d は制約条件第 2 式，第 3 式を満たさなければなりません．例えば $y_s = 1, y_t = 1$ のときは

$$-\beta_s \leq d \leq C - \beta_s$$
$$-\beta_t \leq d \leq C - \beta_t$$

の両式を満たさねばなりません．この場合は $L = \max(-\beta_s, -\beta_t)$, $U = \min(C - \beta_s, C - \beta_t)$ とすれば，制約条件が $L \leq d \leq U$ で与えられます．このような更新と制約を最適化問題に代入して，d の影響しない項を消せば，次の 1 変数 2 次関数の最適化問題になります．

$$\begin{aligned} &\min_{d \in \mathbb{R}} \frac{1}{2}(K_{ss} - 2K_{st} + K_{tt})d^2 - \left(y_s - y_t - \sum_i \beta_i (K_{is} - K_{it}) \right) d \\ &\text{s.t.} L \leq d \leq U \end{aligned} \tag{3.18}$$

この問題は解析的に

$$d = \frac{y_s - y_t - \sum_i \beta_i (K_{is} - K_{it})}{K_{ss} - 2K_{st} + K_{tt}}$$

(ただし $d < L$ のとき $d = L$，$d > U$ のとき $d = U$) と解けます．

各最適化では必ず目的関数の値が変化しないか減少します．例え s, t を一様にランダムに選んだとしても収束します．実用上は，KKT 条件を満たさない α_s を選択してもう片方は適当に α_t を選択します．この選択の方法が SMO アルゴリズムの計算コストを左右しますが，今回の実装ではデタラメに選択することにしています．

3.4 カーネル法

概要のみ説明します．

分離しにくい点達に非線形変換 Φ を施した後に SVM で分類することを考えます．点 x_i，x_j に対して内積 $\langle x_i, x_j \rangle$ が定まっているとし，

$$\langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j)$$

となる関数 k が分かっているとします．SVM では超平面と点との距離を考えて最適化問題を導出しました．距離は点同士の内積で計算できます．非線形変換 Φ を施した点同士の内積は k で計算できるので，Φ で曲げた空間上で SVM による分離問題を考えているの

に実際の計算実行時には Φ は必要ありません．必要なのは適切な k だけです．代表的な k として，多項式カーネルや RBF カーネルがあります．

カーネル法の理論的な側面は [5] 等の本で勉強しようと思います．

3.5 Julia 言語による SMO アルゴリズムの実装

SMO アルゴリズムでは適切に β_s, β_t を選択することが重要ですが，今回は適当に選択しています．打ち切り方法も適当なので最適解に収束しないこともあります．今後，色々な選択方法を試してみたいと思います．

```
1  using LinearAlgebra, Plots
2
3
4  rand_multi_n(n, m_, v__) = randn(n, length(m_)) * v__ .+ m_'
5
6  function svm_testcase2d(n1, n2)
7      x1 = rand_multi_n(n1, [0, 0], diagm([1, 3]))
8      x2 = rand_multi_n(n2, [2, -1], [-1 0; 0 -1])
9      y1 = ones(Int64, n1)
10     y2 = ones(Int64, n2) |> -
11     x = [x1; x2]
12     y = [y1; y2]
13     return x, y
14 end
15
16 function SMO(x, y; k = dot, C = 1, eps = 1e-3)
17     n = size(x, 1)  # データ数
18     p = size(x, 2)  # 各データの次元
19
20     K = [k(x[i,:], x[j,:]) for i in 1:n, j in 1:n]
21     b = zeros(n)
22
23     function optimize(s, t)
24         L = U = 0.0
25         if y[s] == 1
26             if y[t] == 1
27                 L = max(-b[s], b[t] - C)
28                 U = min(C - b[s], b[t])
29             else
30                 L = max(-b[s], b[t])
```

```
31                  U = min(C - b[s], C + b[t])
32              end
33          else
34              if y[t] == 1
35                  L = max(-C - b[s], b[t] - C)
36                  U = min(-b[s], b[t])
37              else
38                  L = max(-C - b[s], b[t])
39                  U = min(-b[s], C + b[t])
40              end
41          end
42
43          numer = y[s]-y[t]-sum(b[i]*(K[i,s]-K[i,t]) for i in 1:n)
44          denom = K[s,s]-2K[s,t]+K[t,t]
45          d = clamp(numer / denom, L, U)
46
47          b[s] += d
48          b[t] -= d
49
50          return d
51      end
52
53      # s, t 選択部分
54      i = 1
55      sz = 8
56      ds = fill(Inf, sz)
57      while sum(ds) > eps
58          s = 1 + mod(rand(Int), n)
59          t = 1 + mod(rand(Int), n)
60          if s == t
61              t = t == n ? 1 : t + 1
62          end
63          ds[i] = optimize(s, t) |> abs
64          i = i == sz ? 1 : i + 1
65      end
66
67      a = b .* y
68
69      w0 = 0  # bias
70      cnt = 0
71      for i in 1:n
72          if eps < a[i] < C - eps
```

```
73                cnt += 1
74                w0 += y[i] - sum(b[j] * K[i,j] for j in 1:n)
75            end
76        end
77        w0 /= cnt
78
79        w = sum(b[i] * x[i,:] for i in 1:n)
80        return x->dot(w, x) + w0, w, w0
81    end
82
83    n1, n2 = 10, 10
84    x, y = svm_testcase2d(n1, n2)
85    f, w, w0 = SMO(x, y, C = 1)
86    svm_plot2d(x, y, f, w, w0) # 関数定義は略
```

ソースコード 3.1　SMO アルゴリズムの実装例

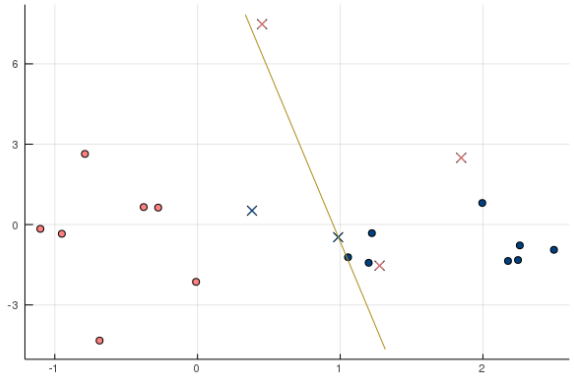


図 3.1　2 クラス分類の例 (1)

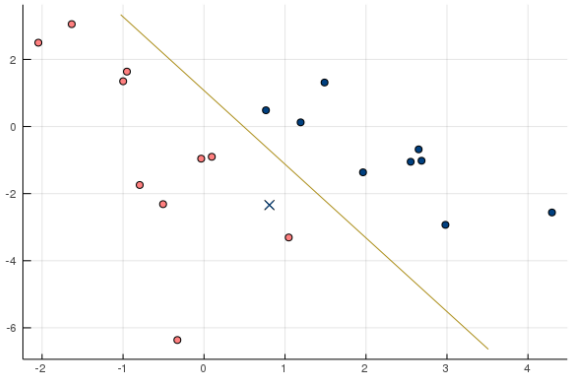


図 3.2　2 クラス分類の例 (2)

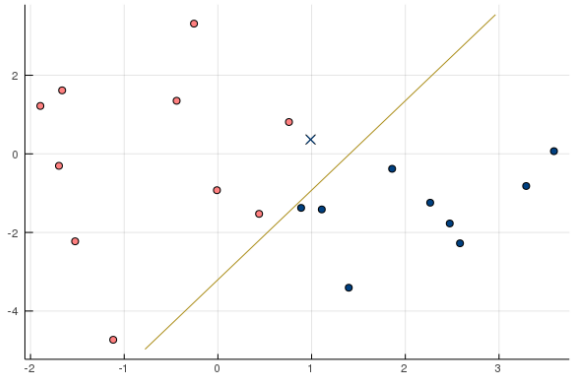


図 3.3　2 クラス分類の例 (3)

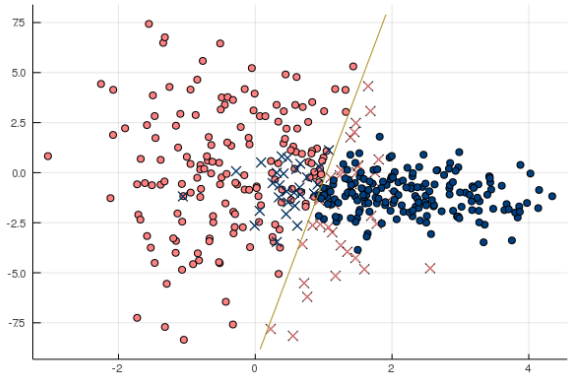


図 3.4　2 クラス分類の例 (4)

3.6　おわりに

まだ理解が追いついていないところが複数あります．強双対定理の周辺は文中に書いた通りですが，双対問題の KKT 点から w_0 を求めるあたりや，カーネル法について丁寧に理解していきたいと思います．

汎用的な制約付き凸 2 次計画問題に対する手法も実装したかったのですが，SVM の最適化問題を解かせると，十数ステップで Inf に発散してしまいました．様子を見ているとステップ幅が大きすぎることが原因のようでした．こういう分野は，数値計算の誤差，アルゴリズムの理解，実装ミス，などの壁が多そうですが，そこが面白そうにも思いました．

気になる点や不備等については，お気軽にご連絡してもらえますと嬉しく思います．最後になりましたが，ここまで読んでいただきありがとうございました．

参考文献

[1] 山下信雄．非線形計画法
[2] 金森敬文，鈴木大慈，竹内一郎，佐藤一誠．機械学習のための連続最適化
[3] 竹内一郎，鳥山昌幸．サポートベクトルマシン
[4] 杉山 将，ほか．統計的学習の基礎
[5] 福水健次．カーネル法入門

4 ラスタースクロールを再現しよう

情報課程 4 回生 上北 裕也

4.1 はじめに

80 年代後半、テレビゲームはまさにその黄金時代を享受していました。

ゲームセンターには FM 音源のメタリックサウンドが溢れ、家庭用ゲーム機では RPG が産声をあげ、小学生は小銭を握りしめ駄菓子屋の筐体に向かい、学校ではゲームの攻略情報に盛り上がる……来るべき 16bit の時代をみんなが心待ちにしていました。今では FM 音源なんてそうそう聞くこともありませんし、RPG 要素もほとんどのゲームに備わっています。駄菓子屋の筐体も見なくなりましたし、小学生は今やスマホでゲームです。それでも今なお在りし日の記憶が、一抹のノスタルジーを伴って、我々をあの頃に振り返らせるのです。私は生まれてないけど。

今回私が紹介するのは、そんな時代の忘れ形見。時代の進歩によって消えてゆく技術を、現代であえて再現する方法の紹介です。

4.2 ラスタースクロールとは？

結論から言うと**画面を揺らしたり、歪ませたりする方法**です。

早速ラスタースクロールの詳しい話に入りたいところですが、そのまま説明するとちょっと混乱するので、まずはブラウン管テレビの液晶画面についてお話します。

たとえば、1 枚の画像をテレビに表示したいとします。テレビはこの画像を小さな点をたくさん並べて作ります。まず、この画像を点の集合と見た時、一番上の行に当たる部分の点をテレビは左から右へ 1 つずつ表示します。一番右の点を表示したら、テレビは左端にもどって 1 つ下の行の点をまたひとつづつ表示します。これを繰り返し、見事最後の行の右端の点を表示すれば晴れて一枚画像が表示できるわけです。

テレビではこの一連の動作をを 60 分の 1 秒 (1 フレーム) に 1 回行い続けています (1

つの点の表示ではなく、1 つの画像の表示ですよ！)。こうして少しずつ画像が変わることによって、われわれの目にはきれいな映像が映し出されるわけなのです。

さて、こうしていそいそ働いているテレビくんに少し意地悪をします。テレビくんが一番右の点を表示して左端に戻っているすきに、次の行から表示する点の位置をすべて 1 つずつ右にずらしてしまうのです！するとテレビくんは何事もなかったかのようにずらした位置に点を表示し続けてしまいます。結果的に意地悪をした行を初めとして、表示される画像は点一つ分右にずれて表示されてしまいます。

さて、ではさらに意地悪を発展させ、点をずらす量を行ごとに少しづつ変えていきます。たとえば、1 行目を 1 つ、2 行目は 2 つ、3 行目は 3 つ……とずらしていくとあら不思議、表示された画像は**平行四辺形に変形**されているのです！この意地悪をもっと発展させ、ずらし方を変えていくと、さらに**複雑な形に画面を変形**させることも出来るのです！

これがラスタースクロールの正体です。用法は様々ですが、80 年代のテレビゲームにはアーケード、家庭用問わず使われていた懐かしの技術です。

4.3　ラスタースクロールの用法

さて、本篇に入る前に少し寄り道をば。先ほど説明したラスタースクロールですが、具体的にどのような場面で使用されていたのでしょう？ここではその一例を紹介します。

4.3.1　多重スクロール

多重スクロールとは、背景の複数のレイヤーを違う速度で動かすことによって立体感を得る技術です。たとえば、流れる雲を表現するとき、画面の上の方にある雲は「近くの雲」なので早く動きます。対して比較的低い位置を流れる雲は「遠くの雲」であるため、遅く流れていくわけです。

多くの場合、この効果を表現するために「レイヤー」と呼ばれる「層」の概念を用います。今回の例であれば、スクロール速度の早い雲と遅い雲を違うレイヤーに配置し、レイヤーごとに別の速度でスクロールさせて処理するのです。他にも背景を毎フレームごとに書き換えるなどの方法がありますが、ラスタースクロールでも多重スクロールを実装することができます。

例えば、背景の上 3 分の 1 を大きく、下 3 分の 1 を小さく、余った真ん中部分を中くらいにずらしつつ、ずらしの大きさをどんどん大きくしていきます。そして背景画像をループさせ、スクロールで空いた部分に背景の出ていった部分を戻してやれば、上段からだんだん速度の変わる多重スクロールとなります。加えて立体感が出るように透過処理を利用

したり、スプライトで画像の付け足しをしたりするのですが、ここでは割愛。

4.3.2　疑似 3D 空間表現

奥行きのある道の画像を円形の線に沿うようにラスタースクロールさせると、道がカーブしているように見えます。これを用いたのがアウトランなどの疑似 3D レースゲームです。これに加えて縦のラスタースクロールを行えば、上り坂や下り坂の勾配の表現も可能となります。

これ以外にも、奥行きのある背景の描画はラスタースクロールの十八番です。スペースハリアーのような疑似 3D 空間の他、手前の地面ほど早くスクロールさせることによって、建物の中における天井と床のスクロール、海や地面などの地平線のある背景などを見事に表現することも可能です。

4.3.3　揺らめくエフェクト

揺らめく炎や蜃気楼などのエフェクトも、ラスタースクロールの得意とするところで、ずらす大きさにサイン波をかけ合わせれば簡単に画像を揺らすことができます。特に有名なところではドラゴンクエストシリーズの旅の扉のうねるような表現がそうです。さらに縦方向のラスタースクロールを併用することによって、疑似的な画面回転、細胞の膨張収縮運動などの表現も可能になるのです！

4.3.4　画面の分割表示

少し変わった用い方ですが、「スクロールさせない」為にもラスタースクロールは用いられます。例えば画面の下部分にスコアやライフなどを表示する場合、上のゲーム画面をスクロールさせつつスコア部分はスクロールさせないようにする必要があります。当時の FC や MSX2 などではこれをラスタースクロールを用いて、分割位置でスクロール量やグラフィックを変えてしまうことによって実装していました。派手な演出だけがラスタースクロールの持ち味ではないのです。

4.4　ラスタースクロールの再現

さて、そんなこんなで大活躍だったラスタースクロールですが、ゲームが 3D 化し、ポリゴングラフィックが主流になっていくと徐々にその姿を消し、今では天然記念物のよう

な扱いを受けています。

今回の目的は、現代においてラスタースクロールを再現することです。流石に画面表示に割り込んでずらす、というのはハードが複雑化した現在では厳しいですが、代わりの方法を用いれば簡単に実装することが可能です。

4.4.1 画像を千切りする

さて、もっとも単純な代替案として登場するのが、「画像の千切り」です。一体どういうことなのかといいますと、背景画像を縦 1 ドットの細いひものように切り分ける、ということです。

もとの通りに並べれば元の画像が復元されますが、ここで「ずらし」をラインごとにかけてやれば見事、ラスタースクロールを再現できるわけです。

ここに、コード例を示します。(C 言語、Dx ライブラリ使用)

```
1  SetTransColor(255, 255, 255);
2  int Road_Handle[HORIZON * 2];
3  int Road_IMAGE = LoadSoftImage("road.bmp");
4  int time = 0;
5  CreateDivGraphFromSoftImage(Road_IMAGE, HORIZON * 2, 1, HORIZON *
       2, 640, 1, Road_Handle);
6  while(1){
7      for (i = 0,time=0; i < HORIZON; i++) {
8          int zure = sin((time+i)*PI/30)*160;
9          DrawGraph(-160 + zure, 160 + i, Road_Handle[i + HORIZON],
               TRUE);
10     }
11     time++;
12     sleep(1000/60);
13 }
```

Road_Handle には千切りにした画像ハンドル (番号) の一つづつが、Road_IMAGE には元画像ハンドルが入ります。time は 1 フレームにつき 1 増える変数です。LoadSoftImage で元画像を読み取り、CreateDivGraphFromSoftImage で分割します。そして、while ループでは 1 フレームごとに進むサイン波を画像のずれに掛け合わせ、波の形を作るようにします。

こうすると、30 フレーム周期で波上に歪む画像ができるわけです。

4.4.2 画像自体を変形させる

画像を波打たせるだけなら、画像自体を変形してしまう、と言うのも手です。Unity での実装例を以下に示します (専門用語多発注意、C#,Unity2D 使用)。

まず、空の (mesh Filter が None の)3D モデルオブジェクトを用意します。2D モードなのに 3D モデル？とお思いかもしれませんが、これは画像を変形させる処理を「テクスチャ (画像) を張り付けた 3D モデルを変形させる」という方法で実装するためです。

用意が出来たら、mesh(3D モデルの骨組みみたいなもの) の頂点を弄るスクリプトを書いてアタッチします。以下にコードの一部を示します。今回重要になってくるのは vertextlist で、これにはメッシュの頂点の情報が格納されています。この情報を毎フレームごとに弄ることによって、滑らかな画像変形を実現します。今回は長方形の画像の縦線の部分に大量の頂点を作り、これらをサイン波に載せるように揺らし、波のような効果を得ます。

```
1  [SerializeField]
2  private MeshFilter meshFilter;
3
4  private Mesh mesh;
5  private List<Vector3> vertextList = new List<Vector3>();
6  private List<Vector2> uvList = new List<Vector2>();
7  private List<int> indexList = new List<int>();
8
9  GameObject director;
10
11 void Start()
12 {
13     mesh = CreatePlaneMesh();
14     meshFilter.mesh = mesh;
15     this.director = GameObject.Find("GameDirector");
16 }
```

```
1  private Mesh CreatePlaneMesh()
2  {
3      var mesh = new Mesh();
4
5      var left = new Vector3(-1, -1, 0);
6      var right = new Vector3(1, -1, 0);
7      vertextList.Add(left);//0番頂点
```

```
8       vertextList.Add(right); //1番頂点
9
10      uvList.Add(new Vector2(0, 0));
11      uvList.Add(new Vector2(1, 0));
12
13      for (var i = 0;i < 100; i++)
14      {
15          left.Set(left.x, left.y + 0.02f, left.z);
16          right.Set(right.x, right.y + 0.02f, right.z);
17          vertextList.Add(left); //2n番頂点
18          vertextList.Add(right);  //2(n+1)番頂点
19
20          uvList.Add(new Vector2(0,(i + 1) * 0.01f));
21          uvList.Add(new Vector2(1,(i + 1) * 0.01f));
22
23          int index = i * 2;
24          indexList.AddRange(new[] { index, index + 2, index + 3,
                index + 1 });
25      }
26
27      mesh.SetVertices(vertextList);//meshに頂点群をセット
28      mesh.SetUVs(0, uvList);//meshにテクスチャのuv座標をセット（今回は割愛）
29      mesh.SetIndices(indexList.ToArray(), MeshTopology.Quads, 0);
            //メッシュにどの頂点の順番で面を作るか
            セット
30      return mesh;
31  }
```

```
1   void Update()
2   {
3       int count = director.GetComponent<GameDirector>().GetDistance
            ();
4       for (var i = 0; i < vertextList.Count; i+=2)
5       {
6           vertextList[i] = new Vector3(-1.0f + i * i * Mathf.Sin(
                count / 60.0f) / 50000.0f, vertextList[i].y, 0);
7           vertextList[i + 1] = new Vector3(1.0f + i * i * Mathf.Sin
                (count / 60.0f) / 50000.0f, vertextList[i+1].y, 0);
8       }
9       mesh.SetVertices(vertextList);
10  }
```

4.5　まとめ

以上、短くはありますがラスタースクロールを再現する種々の方法でした。

ラスタースクロールは、画像変形技術や 3D モデルの技術が簡単に使えるようになった今の世にはほぼ無用の兆物です。しかし、ダライアス外伝の宇宙ステーションが、サマーカーニバル'92 烈火の細胞収縮が、バンパイアキラーの左右に傾く塔が、あるいはガンスターヒーローズのオレンジの乗るヘリの疑似回転が、80、90 年代の素晴らしい演出の数々が私の瞼に焼き付いて離れないのです。

3D モデルはあらゆる表現を可能にしましたが、それによってあらゆる表現が「当たり前のようにできる」と思われてしまう節があると思います。BG 書き換えやパレット変更などにも言えることですが、ラスタースクロールの魅力は、一見不可能な演出を無理やり実装できてしまう、技術という制約の中にあると思います。

長くなりましたが、これで私の記事を締めくくらせていただきます。拙文で申し訳ありませんが、ラスタースクロールの魅力が伝わって下されば幸いです。

編集後記

編集担当の山上です。令和最初の Lime はいかがでしたか？

この Lime は LaTeX によって組版されていますが、当初集まった記事は様々な種類のマークアップ言語で書かれていました。編集作業は難航しましたが、その過程で学ぶこともたくさんあり、貴重な経験を得ることができました。執筆者の皆様には心から感謝します。

最後になりましたが、ここまで目を通していただき、ありがとうございました。

令和元年 11 月 08 日
編集担当　山上 健

Lime Vol. 60

令和元年 11 月 19 日発行 第 1 刷

発行 京都工芸繊維大学コンピュータ部
Web サイト: http://www.kitcc.org/
電子メール: question@kitcc.org